НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Теплоенергетичний факультет
Кафедра автоматизації теплоенергетичних процесів

«На правах рукопису»
УДК 697

«До захисту допущено»
В.о.завідувача кафедри

_____ / *В.А.Волощук*/
“____” _____ 2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності ***151“Автоматизація та комп’ютерно-інтегровані технології”***

на тему: Модель системи автоматизованого керування обігріву двозонного приміщення

Виконав: студент VI курсу, групи ТО-81мп
Тарасюта Дмитро Олексійович _____
(прізвище ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., Волощук В.А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент_____

Київ – 2019 року

Національний технічний університет України
"Київський політехнічний інститут
імені Ігоря Сікорського"

Факультет ***Теплоенергетичний***
Кафедра ***Автоматизації теплоенергетичних процесів***
Рівень вищої освіти – другий(магістерський) за освітньо-професійною програмою

Спеціальність ***151"Автоматизація та комп'ютерно-інтегровані технології"***

ЗАТВЕРДЖУЮ

В.о.завідувача кафедри

_____ /*В.А.Волощук*/
(підпис) (ініціали, прізвище)
" " _____ 2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Тарасюті Дмитру Олексійовичу
(прізвище, ім'я, по-батькові)

1. Тема дисертації Модель системи автоматизованого керування обігріву двозонного приміщення

науковий керівник дисертації Волощук Володимир Анатолійович,д.т.н.,доцент
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «4» листопада 2019 р. № 3812-с

2. Термін подання студентом дисертації «10» грудня 2019 р.

3. Об'єкт дослідження процеси передачі та перетворення енергії, що відбуваються у приміщеннях будівлі з різними теплотехнічними характеристиками та в системі з підтримки теплового комфорту.

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою)
системи, моделі, методи та алгоритми автоматизованого керування створення теплового комфорту в приміщеннях.

5. Перелік завдань, які потрібно розробити
1) Створення моделі системи автоматизованого керування обігрівом приміщення.
2) Розробка SCADA-системи у взаємозв'язку із середовищем Simulink.
3) Порівняння ефективності систем з PID регулятором та термостатом.

4) Розробка стартап-проекту

6. Орієнтовний перелік графічного (ілюстративного) матеріалу
1) Електрична модель термодинаміки приміщення та теплової системи.
2) Схеми обміну даними між SCADA системою та середовищем Simulink.
3) Stateflow діаграма обробки дій користувача в контролері.
4) Зображення бізнес процесів компанії.

7. Орієнтовний перелік публікацій
Тарасюта Д.О., Волощук В.А. “Особливості функціонування “Розумних” теплових мереж”// Матеріали XVII Міжнародної науково-практичної конференції молодих вчених та студентів, м. Київ, 23– 26 квітня 2019 р. У 2 т. – К. КПІ ім. Ігоря Сікорського, 2019. – Т. 2. – 195 с.

8. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання " 04 " вересня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Видача завдання*	04.09.2018	
2	*Аналітичний огляд проблем*	15.11.2018	
3	*Розробка математичної моделі*	25.01.2019	
4	*Розробка комп’ютерної моделі*	14.03.2019	
5	*Створення контролера в Simulink*	20.05.2019	
6	*Порівняння ефективності систем*	18.06.219	
7	*Зв’язок SCADA-системи з Simulink*	01.11.2019	
8	*Розробка SCADA-системи*	15.11.2019	
9	*Стартап-проект*	29.11.2019	
10			
11	*Підпис керівника магістерської дисертації*		
12	*Попередній захист магістерської дисертації*	10.12.2019	
13	*Захист*		

Студент _____ _____
(підпис) (прізвище та ініціали)

Науковий керівник дисертації _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація складається зі вступу, чотирьох розділів, висновку, переліку посилань з 11 найменуваннь, 2 додатків, і містить 77 рисунків, 8 таблиць. Повний обсяг магістерської дисертації складає 129 сторінок, з яких перелік посилань займає 2 сторінки, додатки — 21 сторінку.

Актуальність теми

В наш час весь світ намагається скоротити витрати палива та шкідливих викидів. Система керування є однією із ключових елементів, що необхідні для підвищення енергоефективності. Водночас створення та моделювання систем, що мають логічні операції або містять дії, викликані зовнішніми асинхронними діями, є достатньо складним. Тому в даній роботі створено та реалізовано модель системи керування двозонним приміщенням з використанням бібліотеки Stateflow, що є складовою MATLAB, та має весь необхідний функціонал для створення діаграм стану.

Мета і завдання роботи

Метою даної роботи є створення та реалізація математичної та комп’ютерної моделі автоматизованої системи керування обігріву двозонного приміщення із застосування бібліотек пакету MATLAB та SCADA системи

Об’єкт досліджень – процеси передачі та перетворення енергії, що відбуваються у приміщеннях будівлі з різними теплотехнічними характеристиками та в системі з підтримки теплового комфорту.

Предмет досліджень – системи, моделі, методи та алгоритми автоматизованого керування створення теплового комфорту в приміщеннях.

Наукова новизна отриманих результатів

1. Створено та реалізовано математичну модель теплового режиму будівлі у складі якого містяться приміщення з різними теплотехнічними характеристиками.

2. На основі пакету Stateflow запропоновано підхід до моделювання системи управління обігрівом приміщення як об’єкта зі складною логікою, що забезпечило спрощення створення та реалізацію алгоритму керування.

Практичне значення отриманих результатів

1. Розроблено та реалізовано алгоритм програмного коду в графічному середовищі Stateflow, що є простішим ніж написання в редакторі, та дає можливість зручного відлагодження.
2. Згенеровано C/C++ код зі Stateflow моделі для використання в контролері.
3. Розроблено та реалізовано методику створення SCADA системи на базі моделі в середовищі Simulink через OPC сервер.

Публікації

Тарасюта Д.О., Волощук В.А. “Особливості функціонування “Розумних” теплових мереж”// Матеріали XVII Міжнародної науково-практичної конференції молодих вчених та студентів, м. Київ, 23– 26 квітня 2019 р. У 2 т. – К. КПІ ім. Ігоря Сікорського, 2019. – Т. 2. – 195 с.

Ключові слова

Модель системи керування тепловим режимом будівлі, MATLAB, Stateflow, OPC.

РЕФЕРАТ

Магистерская диссертация состоит из введения, четырех глав, заключения, списка ссылок из 11 наименования, 2 приложений и содержит 77 рисунков, 8 таблиц. Полный объем магистерской диссертации составляет 129 страниц, из которых перечень ссылок занимает 2 страницы, приложения - 21 страницу.

Актуальность темы

В настоящее время весь мир пытается сократить расход топлива и вредных выбросов. Система управления является одной из ключевых элементов, необходимых для повышения энергоэффективности. В то же время создание и моделирование систем, что имеют логические операции или содержащих действия, вызванные внешними асинхронными действиями, есть достаточно сложным. Поэтому в данной работе создано и реализовано модель системы управления двухзонным помещения с использованием библиотеки Stateflow, что является частью MATLAB, и имеет весь необходимый функционал для создания диаграмм состояния.

Цель и задачи работы

Целью данной работы является создание и реализация математической и компьютерной модели автоматизированной системы управления обогрева двухзонного помещения по применению библиотек пакета MATLAB и SCADA системы

Объект исследований - процессы передачи и преобразования энергии, происходящие в помещениях здания с различными теплотехническими характеристиками и в системе по поддержке теплового комфорта.

Предмет исследований - системы, модели, методы и алгоритмы автоматизированного управления создания теплового комфорта в помещениях.

Научная новизна полученных результатов

1. Создан и реализовано математическую модель теплового режима здания в составе которого содержатся помещения с различными теплотехническими характеристиками.
2. На основе пакета Stateflow предложен подход к моделированию системы управления обогревом помещения как объекта со сложной логикой, что обеспечило упрощение создания и реализации алгоритма управления.

Практическое значение полученных результатов

1. Разработан и реализован алгоритм программного кода в графической среде Stateflow, что является простым чем написание в редакторе, и дает возможность удобного отладки.
2. Создан C / C ++ код с Stateflow модели для использования в контроллере.
3. Разработана и реализована методика создания SCADA системы на базе модели в среде Simulink через OPC сервер.

Публикации

Тарасюта Д.А., Волощук В.А. "Особенности функционирования" Умных "тепловых сетей" // Материалы XVIII Международный научно-практической конференции молодых ученых и студентов., Г. Киев, 23-26 апреля 2019 В 2 т. - М.: КПИ им. Игоря Сикорского, 2019. - Т. 2. - 195 с.

Ключевые слова

Модель системы управления тепловым режимом здания, MATLAB, Stateflow, OPC.

ABSTRACT

The master's thesis consists of an introduction, four sections, a conclusion, a list of links of 11 names, 2 appendix, and contains 77 figures, 8 tables. The full volume of the master's thesis is 129 pages, of which the list of links occupies 2 pages, the annexes - 21 pages.

Actuality of theme

Nowadays, the whole world is trying to reduce fuel consumption and emissions. The control system is one of the key elements needed to improve energy efficiency. At the same time, creating and modeling systems containing logical operations or actions caused by external asynchronous actions is quite complicated. Therefore, in this work was created and implemented a model of a two-zone control system using the Stateflow library, which is a component of MATLAB, and has all the necessary functionality to create state diagrams.

The aims and objectives of the study

The aim of this work is to create and implement a mathematical and computer model of an automated control system for two-zone heating using MATLAB libraries and SCADA system.

The object of research is the processes of energy transfer and conversion occurring in the premises of a building with different thermal characteristics and in the system for maintaining thermal comfort.

The subject of research - systems, models, methods and algorithms for automated control of creating thermal comfort in the building.

Innovative novelty

1. A mathematical model of the thermal regime of a building is created and implemented, which contains premises with different thermal characteristics.

2. Based on the Stateflow package, an approach was proposed to model the room heating control system as an object with sophisticated logic, which simplified the creation and implementation of the control algorithm.

The practical significance of the results

1. Developed and implemented a codeflow algorithm in Stateflow graphical environment, which is easier than writing in an editor and allows easy debugging.

2. Generated C / C ++ code from the Stateflow model for use in the controller.

3. Methodology for creation of model based SCADA system in Simulink environment through OPC server was developed and implemented.

Publications

Tarasiuta D.O., Voloshchuk V.A. “Features of Functioning of“ Reasonable ”Thermal Networks” // Proceedings of the XVIII International Scientific and Practical Conference of Young Scientists and Students, Kyiv, April 23-26, 2019 In 2 volumes - K.: KPI them. Igor Sikorsky, 2019. - Vol. 2 - 195 p.

Keywords

Model of control system of building heating, MATLAB, Stateflow, OPC.

ЗМІСТ

ВСТУП

В Європі близько 40% спожитої енергії припадає на потреби будівель, і під цим відсотком обігріву приміщень відповідає понад 75%. Зокрема, середнє споживання на опалення європейських житлових будинків становить 140 кВт*год / $м^2$ на рік, тоді як попит на гарячу та гарячу воду влітку становить приблизно 25 та 20 кВт*год / $м^2$ на рік відповідно. Відсотки не відрізняються суттєво для різних широт в Європі. Крім того, як загальна тенденція (починаючи з 1990 р.), загальне споживання енергії щорічно зростає в розмірі 1%. Дані, надані Європейською Комісією, підкреслюють, що за рахунок підвищення енергоефективності будівель загальне споживання енергії може бути зменшено з до 6%, а викиди CO_2 приблизно на 5%. У 2000-х роках ЄС в запровадили низку директив, стандартів та вказівок, що пропагують стратегії енергозбереження в будівельному секторі, заохочуючи раціональне використання енергії та широке розповсюдження поновлюваних рішень, не порущуючи вимоги комфорту. Зокрема вони визначають інструменти та методи для зменшення використання енергії будівлями та зменшення викидів CO_2 шляхом:

- максимальної експлуатації сприятливих ефектів, пов'язаних із будівельною інерцією та реагуванням на кліматичні умови зовнішнього середовища;
- оптимальне управління мікрокліматом в приміщенні;
- економічно ефективні заходи щодо економії енергії.

Простим прикладом системи опалення будинку, який ще досі поширений, є система де джерелом тепла є електричний нагрівач. Регулятор на основі вкл/викл, є як і не ефективним пристроєм регулювання з точки зору економії ресурсів (наявні реальні експерименти, які показують що ефективність PID регулятора значно вище) так і з точки зору комфорту (відхилення температури приміщення до 2 ^oC).

Система керування є однією із ключових елементів, що необхідні для підвищення енергоефективності. Водночас створення та моделювання систем, що містять дії, викликані зовнішніми асинхронними діями, є достатньо складним. Тому в даній роботі створено та реалізовано модель системи керування двозонним приміщенням з використанням бібліотеки Stateflow, що є складовою MATLAB, та має весь необхідний функціонал для створення діаграм стану.

Також невід’ємною частиною сучасної системи автоматичного керування є SCADA система, що використовується як і в великих промислових об’єктах так і в системах автоматизації житлових будинків. Хороша SCADA система повинна бути обов’язково протестована до її встановлення на реальний об’єкт та добре захищена.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ПРОБЛЕМИ

1.1 Проблема неефективного керування системою опалення на основі включення / виключення

Найпоширенішими пристроями обігріву є радіатори. При розрахунку потреби в енергії в системі обігріву враховується ефекти неідеального управління: фізичні властивості систем опалення та самого регулятора, неточне розташування датчиків. Головними наслідками є нерівномірність температури та плаваюче значення довкола встановленої точки, що призводить до додаткових втрат через стіни будівлі. На сьогодні регулювання температури в будинку, як правило, виконується кімнатними термостатами або термостатичними клапанами на радіаторах, що приводяться в режим відключення, рідко в кращих випадках - PID-логіка для забезпечення бажаної температури в приміщенні і, в той же час, мінімізації явищ перегріву. Термостати не розроблені для оптимальних показників роботи в системах опалення, ні з точки зору економії енергії, ні з точки зору внутрішнього комфорту. По-перше, гістерезис термостата змушує температуру будинку перевищувати значення, на яке встановлений терморегулятор. (Перерегулювання може становити до 2 градусів ^{o}C для типового термостата.) По-друге, типовим джерелом тепла в будинку є нафта або газ (не електрика), а сам обігрівач зазвичай нагріває повітря або воду, що циркулює в будинку. (У випадку з водою або гідравлічними системами вода може циркулювати як перегріта вода або як пара.) У будь-якому випадку теплообмінник, який передає тепло, яке утворюється полум'ям, переносить у воду або повітря має значні втрати тепла. (Хороший котел має ефективність близько 75%, а хороший пленум теплообміну повітря має приблизно таку ж

ефективність.) Один із способів уникнути цих втрат - циркулювати воду чи повітря при температурі, яка змінюється (і, як правило, лише трохи вище температури навколишнього середовища будинку). Ідеальним методом для цього є створення контролера, який змінює витрату теплообмінного середовища. (Змінна швидкість потоку вимагає вентилятора зі змінною швидкістю у разі повітряного теплообміну або насоса зі змінною швидкістю у випадку гідравлічної системи.)

В даній роботі буде використано метод регулювання температури за рахунок зміни витрати теплоносія та PID регулятор, ефективність якого буде порівняно з регулятором, що працює за методом включення / виключення.

1.2 Дослідження ефективності роботи системи для всього опалювального періоду

Для того щоб система автоматизації була дійсно автоматичною, надійною, не потребувала частих коректив потрібно переконатись що вона виконує свої функції у всіх можливих режимах роботи. Тобто в даному випадку система автоматизації опалення приміщення повинна виконувати свої функції протягом всього опалювального сезону. Щоб перевірити працездатність системи розробленої в даній роботі буде проведено моделювання роботи системи в Simulink з використанням метеоданих на весь опалювальний сезон.

Стандарт по будівельній кліматології в Україні встановлює кліматичні характеристики, що використовуються при проектуванні та розрахунках енергоспоживання будівель. Ці параметри визначено на основі спостережень з метеорологічних станцій за 45 років (з 1961 по 2005 рік). До кліматичних параметрів, що надаються стандартом, відносять: значення середньомісячної та середньодобової амплітуди температур, температур найхолоднішої та

найтеплішої доби та п’ятиденки, дати переходу середньодобової температури через 8 та 10ºС восени та весною, параметри вітру, теплової та сонячної радіації, середньомісячної вологості, середньодобової амплітуди вологості та ін.

Цей набір даних дозволяє визначати енергоефективність будівель за рахунок усереднених показників (для розрахунку енергоспоживання) або використовуючи екстремальні точки (для проектування). Використання середніх даних для опалювального сезону або за кожен місяць, в свою чергу, не враховує екстремуми метеорологічних значень параметрів повітря, які спостерігаються протягом року та не дають можливості для динамічного моделювання. Тому варто використовувати річні погодинні дані для різних метеорологічних параметрів, що є репрезентативними для даного клімату. При визначенні енергоефективності будівель в інших країнах досить часто використовуються подібні набори метеорологічних даних. Деякі з них наведені нижче.

Еталонний рік (Test Reference Year або TRY) – один з перших підходів у визначенні щорічних погодинних погодних даних, що спеціально розроблений і використовується в будівництві. TRY’s доступні для 60 місцевостей в США, але вони не включають сонячну радіацію. Даний метод виключає роки з досить високими чи досить низькими середніми за місяць температурами повітря. Для формування TRY використовуються дані з 1948 по 1975 рік.

Набір даних для Типового метеорологічного року (Typical Meteorological Year або TMY) був розроблений з урахуванням недоліків TRY. Період спостережень з 1952 по 1975 роки. Файли даних TMY містять місяці з числа різних років, ці місяці були відібрані на основі щомісячних композитних зважувань вихідних даних. Були вибрані місяці, які розташовувалися ближче до зваженої довготривалої розподілу. Цей набір даних був згодом оновлений до TMY2 на основі нового періоду спостережень (1961-1990) та до TMY3 (1976-2005).

Для представлення одного типового року або групи місяців, ASHRAE впровадила науково-дослідний проект з погодними даними, який названий Погодним роком для енергетичних розрахунків (Weather Year for Energy Calculations або WYEC). Для цього випадку був визначений для кожного місяця року один реальний місяць з погодинними кліматичними даними, які відповідають температурі сухого термометра, яка ближча до середньої температури сухого термометру для цього місяця за 30-річний період спостережень. Обраний місяць перевіряється на наявність нетипових або екстремальних погодних даних по температурі. Окремі дні з інших місяців можуть експортуватися в обраний місяць, якщо їх включення наближує середню температуру місяця до середнього значення за 30-річний період. Дана база була завершена в 1983 році і є в навності для 51 міста в Північній Америці. WYEC була поновлена до WYEC2 у форматі TMY, врахувавши погодинні дані по оцінці хмарності. Оновлена WYEC2 модель враховує компоненти сонячної радіації і дані освітленості.

Також існують і інші бази погодних даних для різних місцевостей, в т.ч. для Європи –«European» Test Reference Year (TRY) і Design Reference Year (DRY), Японії – Automated Meteorological Data Acquisition System (AMeDAS). Процес відбору в «European» TRY дає можливість отримати середнє значення, частоту розподілу окремих змінних і можливі кореляції між різними змінними для кожного місяця в довготривалому періоді даних. Але така база даних, враховуючи температуру сухого термометра, сонячну радіацію і вологість повітря, не враховує швидкість вітру. Місяць з самим меншим відхиленням від середнього включається до бази. DRY є спробою модифікувати TRY з коригуванням обраних місяців. Такі параметри, як температура сухого термометра, сонячна радіація і вологість, але не вітер коригуються шляхом заміни декількох днів днями інших років, але такого ж місяця.

Міжнародна погода для енергетичного розрахунку (International Weather for Energy Calculations або IWEC) - це кліматичні дані, котрі були розроблені в рамках дослідницького проекту ASHRAE RP-1015. Процедура отримання цих даних була заснована на виборі типового року протягом 18-річної послідовності кліматичних даних. Було обрано 12 типових метеорологічних місяців з достатньої кількості інших, шляхом порівняння статистичних показників для кожного місяця з відповідними показниками довготривалої статистики, яка використовує дані денної загальної радіації, температури сухого термометра, температури точки роси і швидкості вітру. Вибір був здійснений на основі зважених коефіцієнтів (0,4 для денної сонячної радіації і 0,3 для температури сухого термометра). База IWEC включає метеодані тільки для двох міст України: м. Києва та м. Одеси. Так як це єдина база, що доступна для умов України та розповсюджується безкоштовно, то в подальшому в дисертації під час моделювання системи керування буде використана саме вона.

На рис. 1.1 зображено погодинний графік зміни температури в опалювальний сезон типового року в м. Києві.

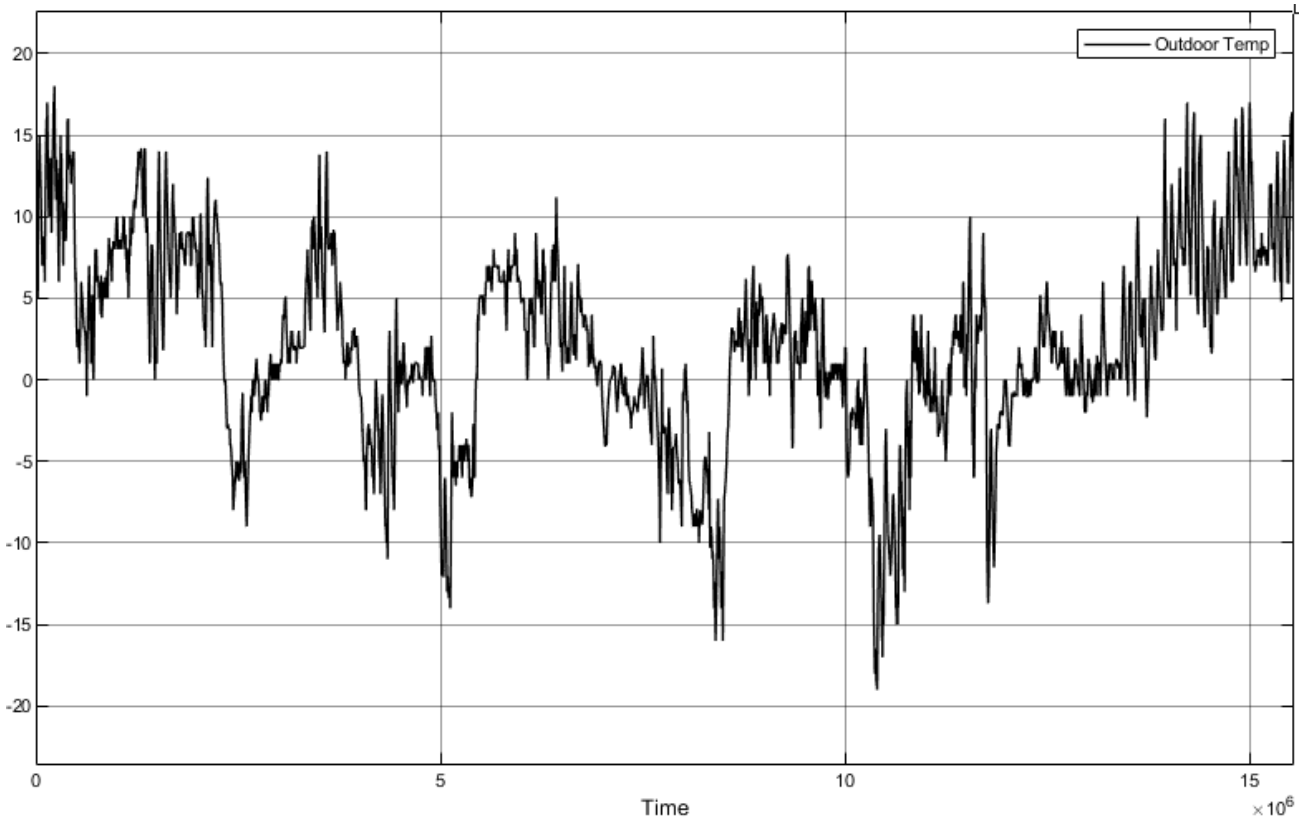


Рис. 1.1 Графік зміни температури повітря в опалювальний сезон типового року в м. Києві

1.3. Створення моделі керування системою зі складною логікою та переходами стану

Моделювання систем, що мають логічні операції або містять дії, викликані зовнішніми асинхронними діями, є складним процесом. Наприклад, під час моделювання дій людини, яка кидає вимикач, або під час моделювання дії, яка змушує комп'ютер обробляти переривання, для моделювання потрібно обробляти подію в момент її виникнення. Коли створюється така модель за допомогою Simulink, блоки та їх з'єднання не завжди дають чіткі вказівки на потік даних, що ускладнює читання та розуміння діаграм. Першою спробою створення візуального зображення складних логічних потоків було представлено Девідом

Харелом у 1980-х роках, візуальний інструмент для представлення всіх можливих станів, які може мати складна реактивна система. Він назвав ідею «діаграмою станів». Основним нововведенням, яке давала діаграма стану, були стани "AND", де частини діаграми могли діяти паралельно з іншими частинами. До впровадження цієї ідеї моделювання кінцевих автоматів спиралося на ідеї Мілі та Мура з 1950-х років. (Таким чином, у підході Харела діаграми стану могли бути активними одночасно - тоді як у підході Мілі та Мура всі стани були “OR” станами.) У 1996 році The MathWorks представила новий інструмент що отримав назву Stateflow, який охоплює багато ідей моделювання та програмування в єдиний і легко зрозумілий інструмент. Назва натякає на те, що інструмент дозволяє моделювати автомати, використовуючи підхід діаграми стану, а також дозволяє графічно представляти потік сигналу. Проектування системи зі складною логікою та переходами стану, які залежать від подій (зовнішніх чи внутрішніх) за допомогою парадигми Stateflow надзвичайно просто.Також одна із суттєвих його переваг це взаємодія із Simulink. Ця особливість дає змогу не тільки розробити логіку для автоматів, але й змоделювати її взаємодію з навколишнім середовищем. Результат - розуміння того, наскільки добре працюють розроблені автомати; тобто дозволяє переконатися, що конструкція відповідає специфікації, оскільки вона працює в симуляції реального світу.

1.3.1 Опис основного функціоналу пакету Stateflow

Stateflow - інструмент для чисельного моделювання систем, що характеризуються складною поведінкою. До числа таких систем відносяться гібридні системи. Прикладами гібридних систем можуть служити системи управління, які використовуються в промисловості (автоматизовані технологічні

процеси), в побуті (складні побутові прилади), у військовій області (високотехнологічні види озброєнь), в сфері космонавтики, транспорту і зв'язку. Всі ці системи складаються з аналогових і дискретних компонентів. Тому гібридні системи - це системи з складною взаємодією дискретної і безперервної динаміки. Вони характеризуються не тільки неперервними змінами стану системи, а й стрибкоподібними варіаціями відповідно до логіки роботи керуючої підсистеми, роль якої, як правило, виконує той чи інший обчислювальний пристрій (кінцевий автомат).

Тож не дивно, що відповідно до зміни оточуючого нас світу змінюються і підходи до його аналізу. Моделювання фізики технологічних процесів (безперервна складова поведінки системи) доповнюється моделюванням логіки роботи керуючих ними пристроїв (дискретна компонента). Математичний апарат опису в даному випадку - це система рівнянь, але не диференціальних, а диференційно-алгебраїчних, для яких відсутній єдиний підхід. Так само йде справа і з наочністю. Візуалізація протікання фізичних процесів забезпечувалася графіками зміни в часі тих чи інших величин. Спроба такого графічного представлення процесів в реактивних системах може закінчитися невдало. Основними причинами цього є багаторазове зростання кількості відображуваних величин і відсутність на графіках інформації про причинно-наслідкові зв'язки між змінними змінними стану.

В даний час для моделювання дискретної динаміки реактивних систем широко використовується запропонований Д. Харелом візуальний формалізм - Statechart (діаграми станів і переходів). Основні неграфічні компоненти таких діаграм - це подія і дія, основні графічні компоненти - стан і перехід.

Подія - щось, що відбувається поза даної системи, можливо вимагаючи деяких дій у відповідь. Події можуть бути викликані надходженням деяких даних

або деяких заданих сигналів з боку людини або деякої іншої частини системи. Події вважаються миттєвими (для обраного рівня абстракції).

Дії - це реакції, що моделюється на події. Подібно подіям, дії прийнято вважати миттєвими.

Стан - умови, в яких модьована система перебуває деякий час, протягом якого вона веде себе однаково. У діаграмі переходів стану представлені прямокутними полями з округленими кутами.

Перехід - зміна стану, зазвичай викликається деякими значимими подіями. Як правило, стан відповідає проміжку часу між двома такими подіями. Переходи показуються в діаграмах переходів лініями зі стрілками, що вказують напрямок переходу. Кожному переходу можуть бути співставлені умови, при виконанні яких перехід здійснюється. З кожним переходом і кожним станом можуть бути співвіднесені деякі дії. Дії можуть додатково позначатися як дії, що виконуються одноразово при вході в стан; дії, що виконуються багаторазово всередині деякого стану; дії, що виконуються одноразово при виході зі стану. Для запобігання ефекту зростання складності при моделюванні великих систем були запропоновані подальші удосконалення. Разом зі станами тепер можуть використовуватися гіперстани (суперстани), що об'єднують кілька станів, що мають ідентичну реакцію на одну і ту ж подію. При цьому замість зображення таких переходів в деякий стан з усіх станів, які охоплюються гіперстаном, зображується тільки один перехід з гіперстаном в зазначений стан (узагальнення переходів). Гіперстани теоретично можуть мати довільну глибину вкладення. Переходи з гіперстану пов'язані з усіма рівнями вкладення. Гіперстани можуть об'єднувати АБО-стани (послідовні стани) і І-стани (паралельні стани). У першому випадку, перейшовши в гіперстан, система може перебувати тільки в одному з станів. У другому випадку система, перейшовши в гіперстан, буде перебувати одночасно в декількох станах.

Діаграми станів і переходів в даний час широко використовуються для моделювання складних систем. Досить згадати уніфіковану мову моделювання (Unified Modeling Language (UML)), одним з елементів якого є діаграми станів і мова "Графсет", який використовуються при програмуванні логічних контролерів систем автоматизації.

Істотно підвищує ступінь наочності моделі використання імітації, що відображає зміни в системі, що супроводжуються переходами від одного стану до іншого. Побудова таких імітаційних моделей можливо з використанням програм Stateflow і Simulink, що входять до складу пакета MATLAB. MATLAB забезпечує доступ до різних типів даних, високорівневому програмування і інструментальних засобів візуалізації. Simulink підтримує проектування безперервних і дискретних динамічних систем в графічному середовищі (у вигляді блок-схем). Stateflow діаграми, що використовують візуальний формалізм Д. Харела, включаються в Simulink-моделі, щоб забезпечити можливість моделювання процесів, керованих подіями. Stateflow забезпечує ясний опис поведінки складних систем, використовуючи діаграми станів і переходів.

Комбінація MATLAB-Simulink-Stateflow є потужним універсальним інструментом моделювання реактивних систем. Додаткова можливість стежити в режимі реального часу за процесом виконання діаграми шляхом включення режиму анімації робить процес моделювання реактивних систем по-справжньому наочним.

Stateflow - потужний графічний інструмент проектування і моделювання комплексних систем локального управління і супервізорного логічного контролю. Використовуючи Stateflow, можна:

- Візуально моделювати комплексні реактивні системи.
- Проектувати детерміновані системи супервізорного управління.

- • Легко змінювати проект, оцінювати результати змін і дослідити поведінку системи на будь-якій стадії проекту.
- Автоматично генерувати програмний код (цілочисельний або з плаваючою точкою) безпосередньо по проекту (для цього потрібно Stateflow Coder).

Stateflow дозволяє використовувати діаграми потоків (flow diagram) і діаграми станів і переходів (state transition) в одній діаграмі Stateflow. Система позначень діаграми потоків - логіка, представлення без використання станів. У деяких випадках діаграми потоків ближче логіці системи, що дозволяє уникнути використання непотрібних станів. Система позначень діаграми потоків - ефективний спосіб представити загальну структуру програмного коду як конструкцію у вигляді умовних операторів і циклів. Stateflow також забезпечує ясний, короткий опис поведінки комплексних систем, використовуючи теорію кінцевих автоматів, діаграми потоків і діаграми переходів станів.

Stateflow робить опис системи (специфікацію) і проект ближче один одному. Створювати проекти, розглядаючи різні сценарії і виконуючи ітерації, набагато простіше, якщо при моделюванні поведінки системи використовується Stateflow. Stateflow використовує варіант системи позначень кінцевого автомата, запропонований Девідом Харела. Діаграма Stateflow - графічне представлення кінцевого автомата, де стани і переходи формують базові конструктивні блоки системи. Ви можете також представляти потокові (що не мають станів) діаграми з використанням Stateflow. Stateflow утворює блоки, які Ви включаєте в модель Simulink. Сукупність Stateflow блоків в моделі Simulink - Stateflow машина.

Додатково Stateflow допускає в представленні ієрархію, паралелізм і хронологію (history). Ієрархія дає можливість організувати комплексні системи, визначаючи структуру об'єкта у вигляді "батьки / нащадки". Тобто ви можете організувати стан всередині інших станів вищого рівня. Система з паралелізмом може мати два або більше активних станів в один і той же час. Хронологія

забезпечує засоби, щоб визначити стан адресата для деякого переходу, грунтуючись на попередній інформації. Ці властивості розширюють корисність даного підходу і відсутні в STDs (State Transition Diagram) і бульбашкових діаграмах.

Кожен блок Stateflow відповідає єдиній діаграмі Stateflow. За допомогою інтерфейсу блок Stateflow підключається до джерел, що надходять від моделі Simulink (дані, події, код користувача).

Stateflow діаграми управляються подіями. Події та дані можуть бути локальними для блоку Stateflow або можуть надходити до і від моделі Simulink і джерел коду, зовнішніх до Simulink.

Ви повинні визначити інтерфейс для кожного блоку Stateflow. Визначення інтерфейсу для блоку Stateflow може включати деякі або всі з цих завдань:

- Визначення методу модифікації блоку Stateflow
- Визначення Output to Simulink (Вихідних до Simulink) подій
- Додавання і визначення нелокальних подій і нелокальних даних в межах діаграми Stateflow
- Визначення відносин з будь-якими зовнішніми джерелами

У наведеному нижче прикладі на рис .1.2 Simulink модель складається з Simulink блоків - Clock, User_Selections і єдиного блоку Stateflow ProcessCommands.

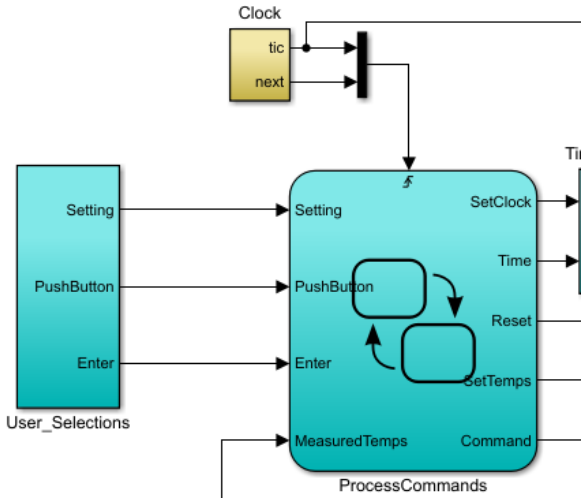


Рис. 1.2 Приклад використання блоку Stateflow

Далі наводиться приклад Stateflow діаграми (рис. 1.3), в якій використовуються основні графічні компоненти. Крім того, детально описуються ці графічні компоненти, а також деякі неграфічні об'єкти і зв'язки між ними.

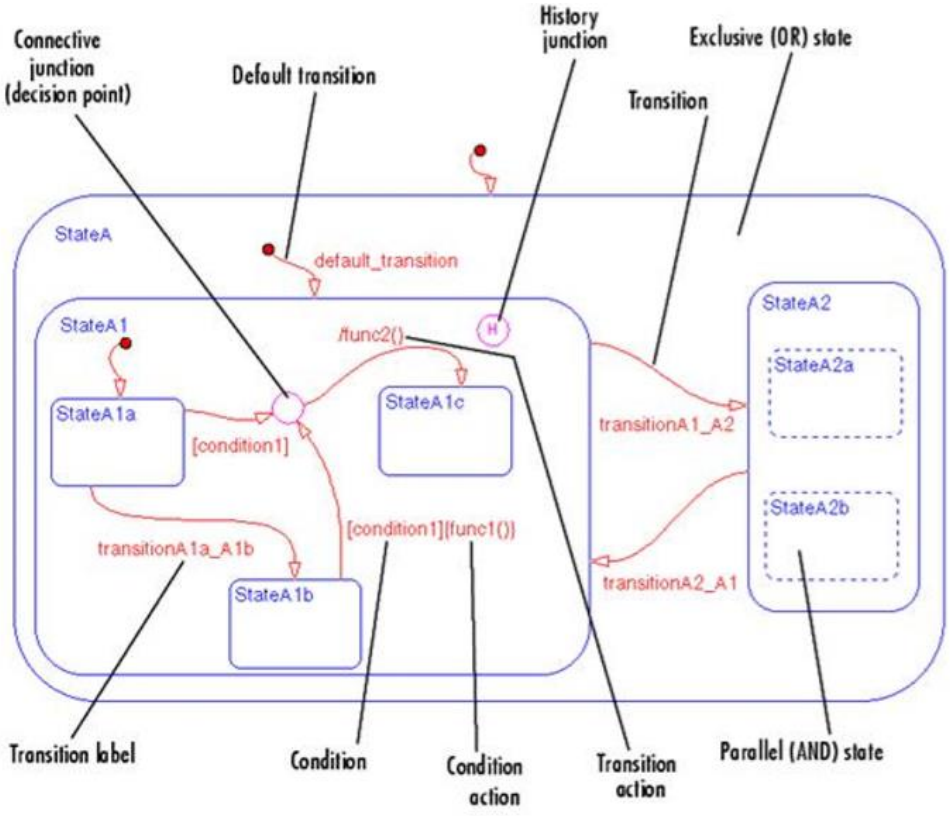


Рис. 1.3 Приклад Stateflow діаграми з основними елементами

Стан (State)

Стан описує режим керованої подіями системи. Динамічні переходи станів від активного до неактивного базуються на події та умові. Кожен стан має батьківський стан. У діаграмі Stateflow, що складається з єдиного стану, батько стану - безпосередньо діаграма Stateflow (також звана коренем діаграми Stateflow). Можна розміщувати стани в межах інших станів вищого рівня. На рис. 1.3 StateA1 - нащадок в ієрархії по відношенню до StateA. Стан також має хронологію. Хронологія забезпечує ефективні засоби базування майбутнього дії на минулій дії. Стани мають мітки, які можуть визначити дії, виконані в послідовності, заснованої на типі дії. Типи дії - entry (на вході), during (протягом), exit (на виході) і on event_name (в разі події з ім'ям _).

Переходи (Transitions)

Перехід - графічний об'єкт, який в більшості випадків пов'язує один об'єкт з іншим. Один кінець переходу прикладений до об'єкта-джерела, а інший кінець - до об'єкта-адресату. Джерело - це місце, де перехід починається, а адресат - місце, де перехід закінчуються. Мітки переходів описують обставини, під дією яких система переходить з одного стану до іншого. Ці обставини - наступ деяких подій, які змушують перехід відбуватися. На рис. 1.3 перехід від StateA1 до StateA2 позначений подією transitionA1_A2, яка змушує перехід статися.

Події (Events)

Події керують виконанням діаграми Stateflow, але є неграфічними об'єктами і таким чином не представлені безпосередньо в діаграмі Stateflow. Всі події, які мають відношення до діаграми Stateflow, повинні бути визначені. Наступ події змушує статус стану (активно - неактивно) в діаграмі Stateflow змінюватися. Наступ події може запускати перехід, і тоді він відбувається, або може запускати дію, і тоді вона виконується. Події наступають по-низхідній, починаючи від батька події в ієрархії.

Події створюються і змінюються за допомогою Stateflow Explorer (Stateflow провідника). Події можуть бути створені на будь-якому рівні ієрархії. Подія має таку властивість, як видимість. Видимість визначає, чи є подія:

- Локальною для діаграми Stateflow
- Входить в Stateflow діаграму від моделі Simulink
- Виходить з Stateflow діаграми в модель Simulink
- Експортується в код, зовнішній до Stateflow діаграми і моделі Simulink
- Імпортується з джерела коду, зовнішнього до Stateflow і Simulink

Дані (Data)

Об'єкти-дані використовуються, щоб зберігати числові значення для застосування в діаграмі Stateflow. Вони є неграфічними об'єктами і таким чином не представлені безпосередньо в діаграмі Stateflow.

Дані створюються і змінюються в Stateflow Explorer. Вони можуть бути створені на будь-якому рівні ієрархії. Дані мають таку властивість, як видимість. Видимість визначає для об'єктів-даних одну з наступних можливостей:

- Бути локальними для діаграми Stateflow
- Входити в Stateflow діаграму від моделі Simulink
- Виходити з Stateflow діаграми в модель Simulink
- Бути тимчасовими даними
- Бути визначиними в робочому просторі MATLAB
- Бути Константами
- Експортуватись в код, зовнішній до Stateflow діаграми і моделі Simulink
- Імпортуватись з джерела коду, зовнішнього до Stateflow і Simulink

Ієрархія

Ієрархія дає можливість організувати складні системи, визначаючи батьківський елемент і структуру об'єктів-нащадків. Ієрархічно побудований проект зазвичай скорочує число переходів і призводить до чітких, зрозумілих

діаграм. Stateflow підтримує ієрархічну організацію як для діаграм, так і для станів. Діаграми можуть існувати всередині інших діаграм. Діаграма, яка існує в іншій діаграмі, називається піддіаграмою.

Аналогічно стани можуть існувати всередині інших станів. Stateflow представляє ієрархію станів з суперстанами і підстанами. Наприклад, діаграма на рис. 1.3 має суперстан, який містить два підстани.

Вихід зі стану вищого рівня або суперстану також має на увазі вихід з будь-яких активних підстанів цього суперстану. Переходи можуть перетинати кордони суперстану, щоб досягти підстан адресата. Якщо підстан активний, його батьківський стан (суперстан) також активний.

Умови (Conditions)

Умова - булевий вираз, що визначає, що перехід відбувається, якщо вказаний вираз є істинним. На рис. 1.3 компонента діаграми [condition1] представляє булевий вираз, який повинний бути істинним, щоб перехід відбувся.

Хронологічні з'єднання (History Junction)

Хронологія - засіб для визначення підстану-адресата переходу по передісторії. Якщо суперстан з виключенням (АБО) декомпозицией має хронологічне з'єднання, підстаном-адресатом при переході буде підстан, відвідуване до цього останнім. Хронологічне з'єднання застосовується до того рівня ієрархії, в якому присутня. Хронологічне з'єднання скасовує будь-які переходи за замовчуванням. На рис. 1.3 хронологічне з'єднання в StateA1 вказує, що, коли перехід до StateA1 відбувається, стає активним той з підстанів (StateA1a, StateA1b або StateA1c), який був активним в останню чергу.

Дії (Actions)

Дії - це результат виконання будь-якої частини діаграми Stateflow. Дія може бути виконана в результаті переходу від одного стану до іншого. Дія може бути також реакцією на стан. На рис. 1.3 сегмент переходу від StateA1b до з'єднання

позначений дією func1 () умови condition 1, а сегмент переходу від з'єднання до StateA1c позначений дією func2 () переходу. Семантика дій буде розглянута пізніше. Перехід, який закінчується в стані, може мати дію умови (condition action) і дію переходу (transition action). Однак переходи, які закінчуються в з'єднаннях, можуть мати тільки дії умов (не допускаються дії переходів). Стани можуть мати дії have entry (на вході), during (протягом), exit (на виході) і on event_name (в разі події з ім'ям _). Наприклад:

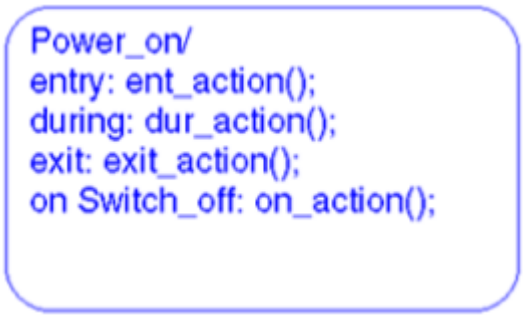


Рис. 1.4 Приклади дій стану

Мова дій визначає типи дій, які Ви можете використовувати і пов'язані з ними системи позначень. Дією може бути звернення до функції, настання події, присвоєння деякого значення змінної і т.д.

Stateflow підтримує парадигми моделювання кінцевого автомата Мура та Мілі. В Мілі моделі дії пов'язані з переходами, в той час як в Мура моделі вони пов'язані з станами. Stateflow підтримує дії станів, дії переходів і дії умов.

Паралелізм

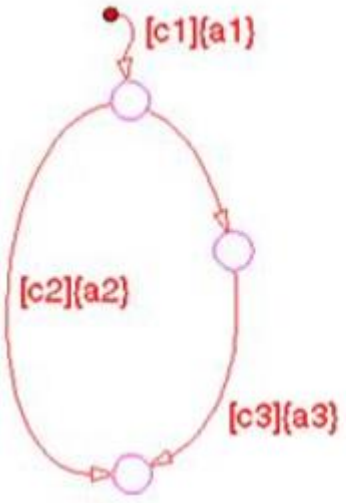
Система з паралелізмом має два або більше станів, які можуть бути активні в один і той же час. Дії кожного паралельного стану по суті незалежні від інших станів. На рис. 1.3 StateA2a і StateA2b - паралельні (I) стану. StateA2 має паралельну (I) декомпозицію стану.

Переходи за замовчуванням (Default Transitions)

Переходи за замовчуванням визначають, яке з декількох виключних (АБО) станів повинно бути активним, коли є невизначеність між двома або більше винятковими (АБО) станами на одному рівні в ієрархії. Наприклад на рис.1.3 перехід за замовчуванням до StateA1 дозволяє неоднозначність, яка існує щодо того, який з підстанів, StateA1 або StateA2, має бути активним, коли суперстан StateA стає активним. У цьому випадку, коли StateA активно, за замовчуванням StateA1 також активно.

З'єднання (Connective Junctions)

З'єднання - точки прийняття рішень в системі. З'єднання - графічний об'єкт, який спрощує Stateflow схематичні представлення і полегшує створення ефективного коду. З'єднання забезпечують альтернативні способи уявити бажану поведінку системи. На представленій на рис. 1.5 Stateflow діаграмі з'єднання використовується як точка прийняття рішення для двох сегментів переходу, що завершуються в стані StateA1c. Наступний приклад показує, як з'єднання (які відображаються у вигляді кіл) використовуються для конструкції if.




```
if [c1]{
     a1
     if [c2]{
        a2
     }else if [c3]{
       a3
     }
}
```

Рис. 1.5 Реалізація оператора if за допомогою елемента З’єднання

Цей фрагмент виконується наступним чином: Якщо умова [c1] істинна, умовна дія a1 виконується і відбувається безумовний перехід до першого (верхнього) з'єднання.

Stateflow визначає, який сегмент переходу верхнього з'єднання вибрати (можна вибрати тільки один). З'єднання з умовами мають пріоритет над з'єднаннями без умов, тобто перехід з умовою [c2] розглядається першим.

Якщо умова [c2] істинна, дія A2 виконується і відбувається перехід до нижнього з'єднання. Так як немає переходу, що виходить з цього з'єднання, виконання діаграми завершено.

Якщо умова [c2] невірна, відбувається безумовний перехід по правому із сегментів (він не має умови).

Якщо умова [c3] істинна, умовна дія a3 виконується і відбувається перехід до нижнього з’єднання. Виконання діаграми завершено.

Якщо умова [c3] невірна, виконання закінчується на середньому з'єднанні.

1.4. Модель керування із застосуванням SCADA системи верхнього рівня автоматизації

Є різні комп'ютерні системи, що використовуються для моніторингу та контролю роботи промислових та не тільки об’єктів, але найбільшого розповсюдження отримала SCADA (Supervisory Control and Data Acquisition) система. Назва системи відображає її основні функції: вона повинна подавати дані, пов'язані з робочим станом системи, і дозволяти операторам віддалено управляти розподіленою системою, що дає можливість операторам ефективно реагувати на зміни в умовах експлуатації та пристосовуватися до змін виробничих цілей.

Зростання поширеності систем SCADA в інфраструктурі є результатом різноманітності переваг, які такі системи можуть надати підприємствам та простим користувачам, які ними користуються. Шляхом переходу від спеціального обладнання до більш гнучких повнофункціональних апаратних засобів, розробка систем управління може бути виконана швидше і з меншими затратами. Є можливість розробки спеціалізованих інтерфейсів системних операторів, що мінімізують труднощі управління та дозволяють швидко та ефективно реагувати на зміни умов процесу. Використання SCADA систем дозволяє на високому рівні керувати промисловим процесом шляхом об'єднання даних із багатьох розподілені частин процесу, що підвищує надійність системи.

Відмінність вимог безпеки SCADA систем та традиційних IT рішень означає що навіть надійні рішення не можуть бути впровадженні без тестування. Щоб запобігти потенційних збитків від несправності та втрати критичної інфраструктури необхідне ретельне тестування. Однак тестування нових рішень для SCADA системи непросто здійснити. Реальні системи явно не можна використовувати через можливі збитки спричинені ненавмисними діями.

Розробка паралельних, але неактивних систем з метою тестування - це підхід, який часто є життєздатним для тестування, але буде надмірно дорогим для тестування складних інфраструктурних установок. Натомість, складність систем SCADA вимагає ретельного моделювання програмного забезпечення, щоб допомогти розкрити переваги та недоліки нових рішень. Моделювання SCADA рішень є складним завданням. Оскільки розробка одноцільового програмного забезпечення для моделювання, що фіксує поведінку лише єдиної системи, було б неефективним та дорогим, моделювання має складатися з простих та багаторазових компонентів, підхід до створення яких повинен бути досить простим та давати можливість використання в інших проектах.

В даній роботі на простому прикладі буде показано один із способів моделювання роботи SCADA системи з симульованим технологічним об’єктом. В якості середовища для моделювання об’єкта (двозонного приміщення) та моделювання системи керування буде використано Simulink, SCADA систему Wonderware InTouch, передача даних з середовища моделювання в SCADA систему буде відбуватись за допомогою OPC сервера.

1.5. Опис технологічної схеми об’єкту управління

Індивідуальний тепловий пункт або ІТП - це комплекс автоматичних пристроїв, зазвичай розташований в підвальній частині будівлі і призначений для того, щоб приєднати внутрішньобудинкові системи теплоспоживання - опалення, гарячого водопостачання або вентиляції - до теплової мережі. Залежно від комплектації теплопункт може управляти системою опалення або системою гарячого водопостачання в будинку, а також керувати обома системами одночасно.

Так як ціллю даної роботи є створення моделі автоматизованого керування обігріву двозонного приміщенні, то для спрощення було обрано одну із типових схем теплового пункту - залежну систему опалення з трьохходовим клапаном. В ІТП з залеж ним приєднанням системи опалення до зовнішніх теплових мереж циркуляція теплоносія в системі опалення підтримується циркуляційним насосом. Управління насосом здійснюється в автоматичному режимі від контролера або від відповідного блоку управління. Автоматичне підтримання необхідного температурного графіка в системі опалення також здійснюється електронним регулятором. Контролер впливає на регулюючий клапан, розташований на трубопроводі, що подає на стороні зовнішньої теплової мережі («гострої воді»). Між подавальним і зворотним трубопроводами встановлено

змішувальну перемичку зі зворотним клапаном, за рахунок якої здійснюється підмішування в подаючий трубопровід з зворотної лінії теплоносія, з більш низькими температурними параметрами.

1.6. Загальна постановка задачі

Отже, на основі проведеного аналізу, досягнення поставленої мети можна забезпечити виконанням наступнх задач:

1. Створення та реалізація моделі системи автоматизованого керування обігрівом двозонного приміщення для всього спектру режимів роботи (протягом всього опалювального сезону) в середовищі Simulink із застосування бібліотеки Stateflow.

2. Розроблення SCADA-системи на основі програмного забезпечення Wonderware InTouch у взаємозвязку із середовищем Simulink через OPC сервер. Відлагодження та тестування роботи системи.

3. Порівняння ефективності систем керування на основі PID регулятора та системи з термостатом з точки зору створення комфорту та споживання енергоносія.

4. Розробленя стартап-проекту із ефективності створення аглоритмів керування системам теплозабезпечення на основі бібілотеки Stateflow.

РОЗДІЛ 2
МАТЕМАТИЧНА ТА КОМП'ЮТЕРНА МОДЕЛЬ ОБ'ЄКТУ

2.1 Математична модель об'єкту керування

У наступному розділі буде розроблено регулятор обігріву, який замінює терморегулятор, із наступними перевагами:

• кращий контроль температури будинку,

• менше споживання енергії для того ж або кращого рівня комфорту,

• значно краще відчуття мешканців.

Шлях до досягнення цих цілей полягає в ретельному розумінні динаміки теплового потоку в приміщенні, тому потрібно детально змоделювати приміщення. Зважаючи на це, побудуємо модель для обігріву двохзонного приміщення в Simulink та запустимо її для розуміння проблем. Нехай температура приміщення буде T (x, y, z, t), де три ортогональні просторові напрямки x, y, z відносно деякої системи координат, а температура є функцією від часу так як тепло подається через систему обігріву. Основним рівнянням для обчислення температури є рівняння теплопровідності:

$$\rho c_p \frac{\partial T(x,y,z,t)}{\partial t} = \nabla(k(x,y,z,t)\nabla T(x,y,z,t) + Q(x,y,z,t)) \quad (2.1)$$

Джерелом тепла в приміщеннях є конвектори (радіатори), які розташовані вздовж стін на підлозі, залежно від форм приміщення. Ці джерела тепла - це вхідне тепло Q (x, y, z, t).

2.1.1 Кінцеві просторові моделі

Є багато варіантів спрощень, які ми можемо використовувати для моделювання системи обігріву приміщення за допомогою даного рівняння.

Спочатку ми припускаємо, що ділянки стін, стелі, підлоги та скла мають постійну теплопровідність (тобто k буде постійною для кожного з різних типів поверхні). По-друге, ми можемо припустити, що кожна кімната в нашому будинку - це окрема об'ємна сутність, так що тепло, що зберігається в кожній кімнаті ($\rho c_p \frac{\partial T(x,y,z,t)}{\partial t}$ в рівнянні теплопровідності) незалежне для кожної з кімнат. Останнє припущення, яке ми можемо зробити, - це те, що потік тепла в приміщеннях і поза ними (через стіни, стелі тощо) призведе до лінійного теплового градієнта в матеріалі.

Можна також припустити, що теплообмінники мають деяку теплоємність (тобто вони можуть зберігати тепло протягом деякого часу). Кожен теплообмінник передаватиме своє тепло в приміщення з деякими тепловими втратами. Цей теплообмін відбувається через межі повітря, які, хоча і не проводять тепло (теплообмін фактично відбувається за допомогою конвекції), моделюються, передбачаючи лінійний градієнт у повітрі навколо обмінника. Також робимо важливе припущення, що тепло, що виходить з дому, не змінює температуру зовнішнього повітря. Також припускаємо, що процес горіння є постійним джерелом тепла з відповідними показниками ефективності горіння; це означає, що кожен літр нафти або газу виробляє постійну кількість тепла. Згідно з цими припущеннями, рівняння тепла стає набором взаємопов'язаних елементів, що складаються з окремих приміщень та окремих теплообмінників.

Ці припущення - постійні температури зовнішнього повітря, лінійні градієнти в стінках та навколо теплообмінників та постійний потік тепла від процесу горіння в середовищі, що використовується для теплообміну - означають, що тепловий потік у стабільному стані з найтепліших частин будинку до холоду зовні постійний. Це випливає з того, що з цими припущеннями вираз $k(x,y,z,t)\nabla T(x,y,z,t)$ є постійним для кожної з кімнат.

Моделюючи однокімнатне приміщення, можна просто описати модель як диференційне рівняння першого порядку для всього будинку. Оскільки будинок був єдиним цілим, температура кожної кімнати в будинку була однаковою. Очевидно, що це не так у типовому приміщенні. Термодинаміка кожної кімнати взаємопов'язана, але також є незалежною, оскільки втрати тепла від проникнення, провідності та випромінювання залежать лише від матеріалів у стінах, кількості вікон та дверей та простору, що оточує приміщення (підвали, горища, інші кімнати, сонячні простори тощо).

У загальному наближенні параметра окремі температури приміщення однакові у всьому об'ємі приміщення. Ці припущення також означають, що чисте тепло, що надходить у відсік або виходить із нього (обсяг простору, який має постійну теплоємність та рівномірну температуру), дорівнює нулю. З рівняння тепла, коли різниця градієнта дорівнює нулю, немає чистого підвищення температури відсіку від цього потоку - і значення лінійного градієнту підтверджує, що це правда. Ми побачимо, що це припущення еквівалентно тому, що в електричному ланцюзі струми, що впадають у будь-який вузол і виходять із нього, повинні дорівнювати нулю.

Таким чином, для двокімнатного будинку рівняння теплового потоку розділяється на чотири окремих звичайних диференційних рівняння: два для кімнатної температури та два для теплообмінника.

2.1.2 Електричний аналог рівнянь теплових режимів обєкту

Після того, як ми «зібрали» кімнати та обігрівачі в єдині об'єкти і зробили припущення про те, як рухається тепло, рівняння тепла стає аналогічним електричному ланцюгу з резисторами, конденсаторами, джерелами струму та джерелами напруги. Аналоги різних значень елемента рівняння тепла такі:

- Потік тепла аналогічний струму (а джерело струму - джерело тепла).
- Температура елемента аналогічна напрузі в ланцюзі (і тому джерело напруги - це постійна температура, незалежно від тепла, що втікає або витікає).
- Теплопровідність матеріалу аналогічна резистору (аналогія насправді з провідністю, тобто $1 / R$ - теплопровідність, що буде важливо, коли нам потрібно поєднувати шляхи теплового потоку).

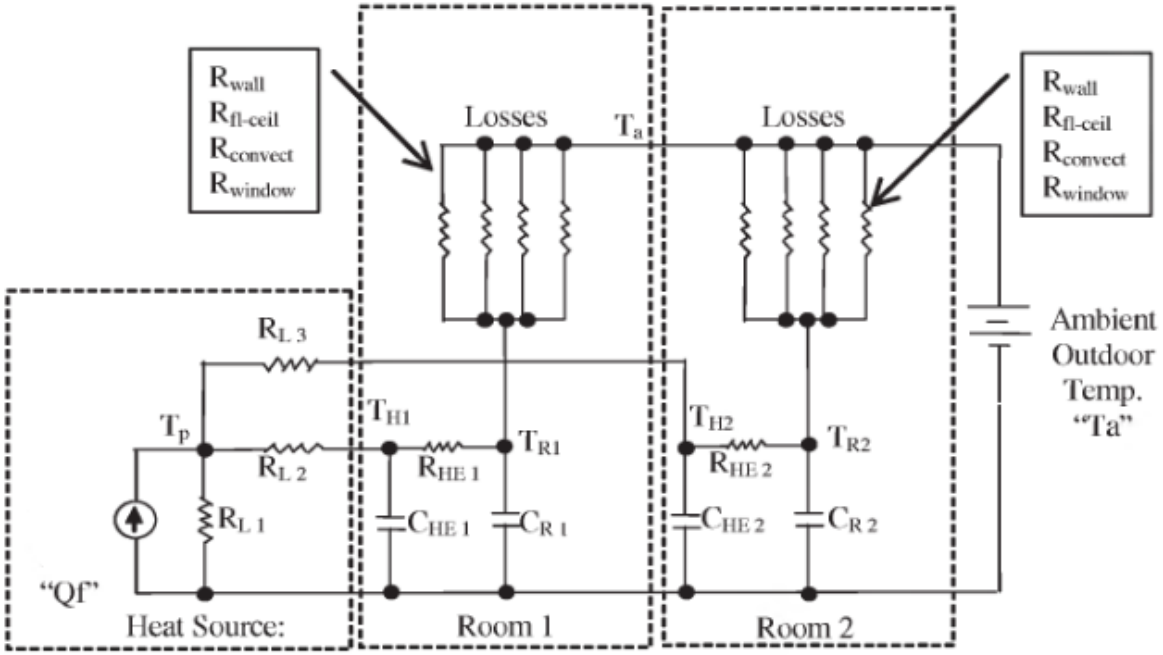


Рис. 2.1 Електрична модель термодинаміки приміщення та теплової системи

Теплоємність, помножена на об'єм елемента, $\rho c_p \Delta V$, аналогічна електричній ємності. Використовуючи цю аналогію, маємо наступне:

- Окремі кімнати в приміщенні можна моделювати як єдиний конденсатор.
- Тепловий потік між приміщеннями і зовні моделюється як струм, що протікає через резистори.
- Теплообмінники приміщення також моделюються як конденсатори.

- Теплові втрати від джерела тепла до теплообмінників також можна моделювати як резистори.
- Джерело тепла (теплопункт) є джерелом струму.
- Зовнішня температура (постійна) є джерелом напруги.

Взявши все це до уваги, можемо моделювати наш двохзонне приміщення як електричний ланцюг зображений на рисунку 2.1.

Цю модель легко зрозуміти, оскільки вона відображає (зібрану) динаміку що лежить в основі. Має бути очевидним, що модель в даному випадку - це не рівняння (у тому сенсі, що модель Сімулінка явно представляє диференціальні рівняння), але ми можемо пройти кожен фрагмент у цій схемі, щоб зрозуміти, що саме воно має на увазі в термінах термодинаміки.

Отже, з рисунку 2.1, отримано наступне:

- Кожна кімната і кожен теплообмінник представлені конденсатором (це теплоємність приміщення або теплообмінника, задана як ρcV). Для кожного приміщення це добуток густоти повітря, питомої теплоти повітря та об'єму приміщення, а для кожного теплообмінника - це добуток густини теплообмінного середовища (води чи повітря), питома теплоємність цього матеріалу та об'єм теплообмінника.
- Тепло, що надходить у приміщення або виходить із нього, залежить від теплопровідності, k, яку перетворимо в тепловий опір за допомогою $1 / k$. Таким чином, чотири паралельних резистори, у кожному з приміщень, є шляхами теплового потоку через теплові втрати від стін, стель, конвекції повітря ззовні всередину та вікон. Вони проводять тепло (струм в даному випадку тепло) з приміщення до фіксованої зовнішньої температури, яка є джерелом напруги (Ambient Outdoor Temp).

- Потік тепла від теплообмінників до приміщень передбачає деяке зниження температури середовища, тому приміщення не нагрівається так, як теплообмінники (резистори $R_{HE1,2}$).
- Тепло від джерела тепла повністю не забирається. (Процес передачі тепла від котла до теплообмінника включає втрати, які моделюються як резистори $R_{L1,2,3}$)
- Перевага даного електричного аналога рівняння тепла полягає в тому, що легко змінити модель, щоб зробити її більш складною і включати більше ефектів.

2.2 Перетворення кінцевої моделі в рівняння для моделювання в Simulink

Електричне коло зображене на рис. 2.1 моделює теплові потоки і температури в приміщеннях, але не у формі диференціальних рівнянь. Для перетворення схеми в набір диференціальних рівнянь потрібно використати закон Кірхгофа, який говорить, що сума всіх струмів, що входять у будь-який вузол електричного кола, повинна дорівнювати нулю. Для цієї моделі, температури для кожного приміщення та теплообмінників (напруги конденсатора) є "вузлами". Зовнішні джерела тепла та радіатори - це джерела напруги та струму, які представляють "постійні" температури навколишнього середовища та тепловий потік. (Вони не є постійними протягом тривалого інтервалу часу, але протягом інтервалу часу, яким оперує система обігрів, вони є сталими.) Результатом цих маніпуляцій на тепловому рівнянні є рівняння, що використовуються в моделі Simulink.

Ми зафіксували важливі особливості будинку для проектування системи обігріву без явного вирішення теплового диференціального рівняння з

частинними похідними. Метод, який ми тут використали, є лише одним із способів перетворення диференціального рівняння з частинними похідними у зв'язаний набір диференціальних рівнянь. Всі вони називаються "моделями з кінцевими елементами". У даному випадку - це великі маси, які мають більш-менш постійні температури (кожне приміщення та кожен теплообмінник).

2.2.1 Застосування закону Кірхгофа для отримання рівнянь

Рівняння струму Кірхгофа в кожному з ємністних вузлів моделі дає нам диференційне рівняння для моделювання. Таким чином маємо наступне:

Для теплообмінника 1:

$$C_{HE1}\frac{dT_{H1}}{dt} = -\left(\frac{1}{R_{HE1}} + \frac{1}{R_{L2}}\right)T_{HE1} + \frac{1}{R_{HE1}}T_{R1} + \frac{1}{R_{L2}}T_p \tag{2.2}$$

Для кімнати 1:

$$C_{R1}\frac{dT_{R1}}{dt} = -\frac{1}{R_{eq1}}T_{R1} + \frac{1}{R_{HE1}}T_{H1} + \frac{1}{R_{eq2}}T_a \tag{2.3}$$

Де

$$\frac{1}{R_{eq1}} = \left(\frac{1}{R_{wall1}} + \frac{1}{R_{f1-ceil1}} + \frac{1}{R_{convect1}} + \frac{1}{R_{window1}} + \frac{1}{R_{HE1}}\right) \tag{2.4}$$

$$\frac{1}{R_{eq2}} = \left(\frac{1}{R_{wall1}} + \frac{1}{R_{f1-ceil1}} + \frac{1}{R_{convect1}} + \frac{1}{R_{window1}}\right) \tag{2.5}$$

Для теплообмінника 2:

$$C_{HE2}\frac{dT_{H2}}{dt} = -\left(\frac{1}{R_{HE2}} + \frac{1}{R_{L3}}\right)T_{H2} + \frac{1}{R_{L3}}T_{R2} + \frac{1}{R_{L3}}T_p \tag{2.6}$$

Для кімнати 2:

$$C_{R2}\frac{dT_{R2}}{dt} = -\frac{1}{R_{eq3}}T_{R2} + \frac{1}{R_{HE2}}T_{H2} + \frac{1}{R_{eq4}}T_a \tag{2.7}$$

Де

$$\frac{1}{R_{eq3}} = \left(\frac{1}{R_{wall2}} + \frac{1}{R_{f1-ceil2}} + \frac{1}{R_{convect2}} + \frac{1}{R_{window2}} + \frac{1}{R_{HE2}}\right) \tag{2.8}$$

$$\frac{1}{R_{eq4}} = (\frac{1}{R_{wall2}} + \frac{1}{R_{f1-ceil2}} + \frac{1}{R_{convect2}} + \frac{1}{R_{window2}}) \quad (2.9)$$

Існує також алгебраїчне (не диференціальне рівняння) для температури теплообмінного середовища, T_p, задане як сумма теплового потоку в цей вузол при джерелі тепла.

Для теплового потоку до теплообмінників:

$$T_p = \frac{Q_f + \frac{T_{H1}}{R_{L2}} + \frac{T_{H2}}{R_{L3}}}{(\frac{1}{R_{L1}} + \frac{1}{R_{L2}} + \frac{1}{R_{L3}})} = R_{eq5}(Q_f + \frac{T_{H1}}{R_{L2}} + \frac{T_{H2}}{R_{L3}}) \quad (2.10)$$

Де

$$\frac{1}{R_{eq5}} = (\frac{1}{R_{L1}} + \frac{1}{R_{L2}} + \frac{1}{R_{L3}}) \quad (2.11)$$

Підстановка цього алгебраїчного результату в попередні диференційні рівняння вище дає остаточну форму цих двох рівнянь як

$$C_{HE1}\frac{dT_{H1}}{dt} = -\left(\frac{1}{R_{HE1}} + \left(1 - \frac{R_{eq5}}{R_{L2}}\right)\frac{1}{R_{L2}}\right)T_{H1} + \frac{1}{R_{HE1}}T_{R1} + \frac{R_{eq5}}{R_{L2}R_{L3}}T_{H2} + \frac{R_{eq5}}{R_{L2}}Q_f \quad (2.12)$$

$$C_{HE2}\frac{dT_{H2}}{dt} = -\left(\frac{1}{R_{HE2}} + \left(1 - \frac{R_{eq5}}{R_{L3}}\right)\frac{1}{R_{L3}}\right)T_{H2} + \frac{1}{R_{L3}}T_{R2} + \frac{R_{eq5}}{R_{L2}R_{L3}}T_{H1} + \frac{R_{eq5}}{R_{L3}}Q_f \quad (2.13)$$

Ці диференційні рівняння було досить легко розписати, оскільки було лише чотири диференційних рівняння (і одне алгебраїчне рівняння); але, якби потрібно було змоделювати будинок з 15 кімнатами, це завдання було б досить важким. При розробці цих рівнянь ми визначили деякі еквівалентні резистори. Ці еквіваленти мають сенс з точки зору втрат тепла для приміщень та теплообмінників. Комбінована втрата для кожної кімнати - це еквівалентний резистор, який отримали розмістивши резистори для втрат паралельно, тому чисті втрати - це комбінований ефект тепла, що протікає через стіни, перекриття підлоги, двері та вікна та провідний шлях теплообмінників. На практиці ці опори обумовлюють конвективні втрати, що зумовлені проникненням повітря в приміщення. Оскільки проникнення повітря залежить від зовнішніх вітрів, ці

додаткові умови включають швидкість вітру. Ми проігнорували цей ефект у нашій моделі.

2.2.2 Модель в просторі станів

Тепер, коли у нас є рівняння для моделювання системи обігріву будинку, ми створимо модель в просторі станів для використання в Simulink. Припустимо, що напруга (температура) кожного конденсатора - це стан у моделі чотирьох станів (одне диференціальне рівняння або стан для кожної температури). Рівняння вище стають:

$$C\frac{d}{dt}\begin{bmatrix} T_{R1} \\ T_{R2} \\ T_{HE1} \\ T_{HE2} \end{bmatrix} = G\begin{bmatrix} T_{R1} \\ T_{R2} \\ T_{HE1} \\ T_{HE2} \end{bmatrix} + B_1 T_a + B_2 Q_f \tag{2.14}$$

Вектор станів буде $\boldsymbol{x(t)}$, і можна інвертувати матрицю $\boldsymbol{C}$ щоб отримати модель в просторі станів,

$$\frac{d}{dt}x(t) = C^{-1}Gx(t) + C^{-1}B_1 T_a + C^{-1}B_2 Q_f \tag{2.15}$$

Матриця $\boldsymbol{C}$ є діагональною, тому її обернена є просто зворотною її елементами, і тому кінцева форма рівнянь складається з таких матриць:

$C^{-1}G$

$$= \begin{bmatrix} -\frac{1}{C_{R1}R_{eq1}} & 0 & \frac{1}{C_{R1}R_{HE1}} & 0 \\ 0 & -\frac{1}{C_{R2}R_{eq3}} & 0 & \frac{1}{C_{HE1}R_{HE1}} \\ \frac{1}{C_{HE1}R_{HE1}} & 0 & -(\frac{1}{C_{HE1}R_{HE1}} + (1-\frac{R_{eq5}}{R_{L2}})\frac{1}{C_{HE1}R_{L2}}) & \frac{R_{eq5}}{C_{HE1}R_{L2}R_{L3}} \\ 0 & \frac{1}{C_{HE2}R_{HE2}} & \frac{R_{eq5}}{C_{HE2}R_{L2}R_{L3}} & -(\frac{1}{C_{HE2}R_{HE2}} + (1-\frac{R_{eq5}}{R_{L3}})\frac{1}{C_{HE2}R_{L3}}) \end{bmatrix} \tag{2.16}$$

$$C^{-1}B_1 = \begin{bmatrix} \frac{1}{C_{R1}R_{eq2}} \\ \frac{1}{C_{R2}R_{eq4}} \\ 0 \\ 0 \end{bmatrix} \qquad C^{-1}B_2 = \begin{bmatrix} 0 \\ 0 \\ \frac{R_{eq5}}{C_{HE1}R_{L2}} \\ \frac{R_{eq5}}{C_{HE2}R_{L2}} \end{bmatrix} \tag{2.17}$$

Цей набір рівнянь є основою для Simulink моделі системи обігріву. Для завершення моделі потрібні числові значення параметрів.

Для визначення параметрів потрібно визначити розміри приміщення. Припустимо, що кімната 1 прямокутної форми 6 * 9 * 2.5 метри; таким чином площа підлоги та стелі становить 135 квадратних метри. Кімната 1 має чотири вікна та одні двері, загальною площею 7 квадратних метрів. Кімната 2 також є прямокутною розміром 6 * 12 * 2.5 метри. Припускаємо, що дві кімнати суміжні вздовж 6-метрового розміру і що тепловий потік через цю стіну є незначним (і ми проігноруємо це). Кімната 2 має вісім вікон та дві двері, спільна площа яких становить 17 квадратних метрів. Таким чином, площа відкритих стін у приміщенні 1 становить загальну площу поверхні за мінусом скла та дверей, яка становить 53 квадратних метрів. Аналогічно, кімната 2 має відкриту поверхню (мінус скло та двері) площею 58 квадратних метри.

Підлоги та стелі обох кімнат мають 30 сантиметрів склопластикової ізоляції теплопровідність якої $1/3$ ККал / годину на квадратний метр площі поверхні підлоги та стелі на градус різниці температур кімнати та зовні. Стіни кожної кімнати також були утеплені склопластиком теплопровідність яких рівна 1/1.4 ККал / годину на квадратний метр ізольованої площі стіни та на градус різниці температур.) Останнє еквівалентне значення теплопровідності, яке нам потрібно, - це вікна та двері. З подвійним склом, більшість сучасних вікон (та дверей) мають теплопровідність рівну 1/0.3 (ті ж одиниці, що і вище). В усіх цих числах ми припускаємо, що конвекція була включена в еквівалент R, тому не використовували термін еквівалентних опорів для конвекції (проте ми дослідимо

ефект додаткових втрат на конвекцію за допомогою цього терміна). Для розрахунку всіх значень теплового опору вище, значення наведені за годину, але моделювання буде в секундах, тому дані потрібно перевести.

Система обігріву використовує воду як теплообмінне середовище. Теплопровідність (та що позначено опором), що сполучає теплообмінники в приміщеннях, становить 0.1 ККал / годину на лінійний метр теплообмінника на градус С різниці температур між приміщеннями та теплообмінниками. Ми припускаємо, що в кімнаті 1 теплообмінник площею 2.3 метри, а в кімнаті 2 – 2.8 метри. Ми також припускаємо, що втрати в процесі теплообміну становлять 0,05.

Питома теплоємність повітря становить 0,24 ККал на кілограм на градус С, а густина повітря – 1.293 кілограми на кубічний метр.

Комбінуючи ці величини разом, ми отримуємо наступні значення для компонентів моделі в просторі станів:

Теплоємність приміщення 1 - це маса повітря в приміщенні, помножена на питому теплоту повітря. Об'єм приміщення - 135 куб. метрів, тому маса повітря - 135 x 1.293 = 174 кілограми, а теплоємність - 0,24 * 174 = 41.76 ККал на градус С. Аналогічно, ємність приміщення 2 – 55.85 ККал на градус С. Для теплообмінників ми вважаємо, що об'єм води в них дорівнює 0.015 кубічного метра. Оскільки питома теплоємність води дорівнює 1, то теплова ємність кожного з теплообмінників становить 43,02 ККал на градус С. Таким чином, ємності, що використовуються в моделі (в ККал / градус С) є

$$C_{R1} = 41.76, \qquad C_{R2} = 55.85, \qquad C_{HE1} = C_{HE2} = 43.02$$

Теплові опори для моделювання в різних частинах будинку не є числами термічного опору в диференціальних рівняннях (розмірності різних величин показують це). Для отримання цих чисел нам потрібно помножити числа провідності для різних поверхонь на їх площі. Таким чином, ми маємо (до переведення з годин у секунди)

$$R_{HE1} = R_{HE2} = R_{HE3} = 0.2 * 20 = 0.4$$

$$R_{L1} = R_{L2} = R_{L3} = 0.05$$

$$R_{wall1} = \frac{1.4}{53} = 0.0264, \qquad R_{f1-ceil1} = 2 * \frac{3}{60} = 0.1, \qquad R_{window1} = \frac{0.3}{7} = 0.0417$$

$$R_{wall2} = \frac{1.4}{58} = 0.0242, \qquad R_{fl-ceil2} = 2 * \frac{3}{75} = 0.08, \qquad R_{window2} = \frac{0.3}{17} = 0.017$$

2.2.3 Компютерна модель об’єкту керування в середовищі Simulink

Створена нами модель в Simulink використовує модель в просторі станів та використовує можливість векторизації Simulink. Модель, отримана в результаті виконаних операцій, зображена на рис.2.2. Ця модель працюватиме незалежно від того, скільки приміщень та теплообмінників ми моделюємо.

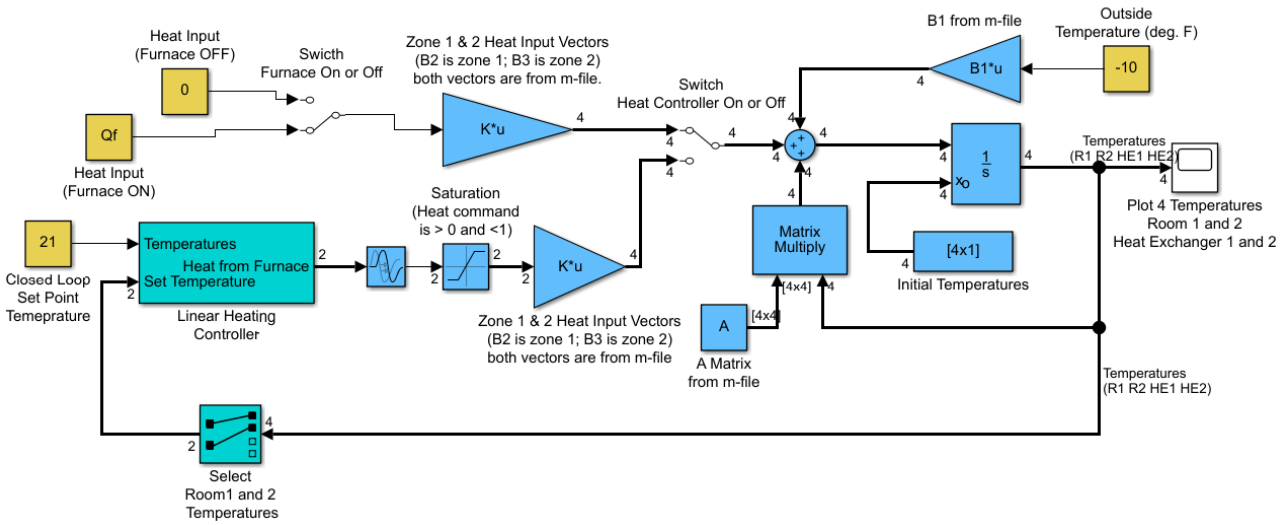


Рис. 2.2 Simulink модель системи керування опаленням двозонного приміщення

На рис. 2.3 – 2.13 зображено кожен елемент системи окремо з відповідним детальним описом функції в системи.

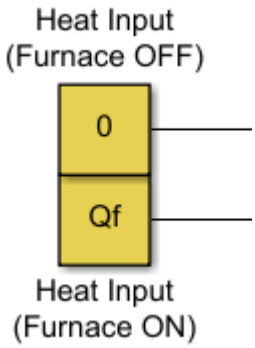


Рис. 2.3 Джерела тепла для моделювання (вкл. та викл.)

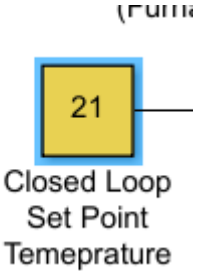


Рис. 2.4 Блок задання температури

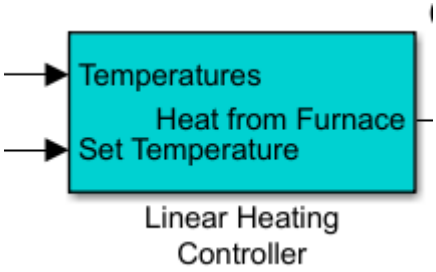


Рис. 2.5 Блок ПІД регулятора

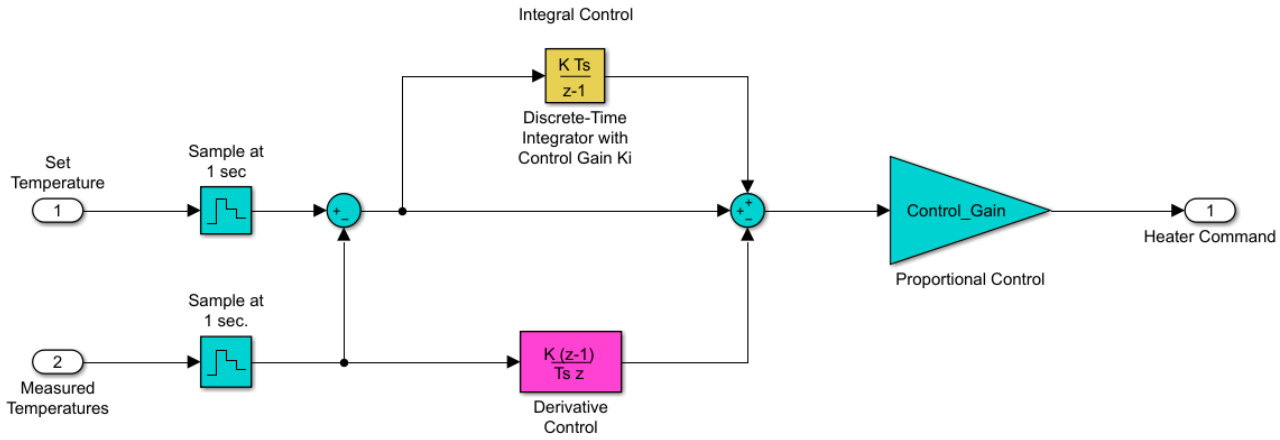


Рис. 2.6 Структура ПІД регулятора

Де блок Sample at 1 sec використовуэться для імітації часу опитування контролера що дорівнює 1 секунда. Integral Control та Derivative Control блоки інтегральної та диференціальної дискретної складової. Proportional Control – пропорцыйна складова.

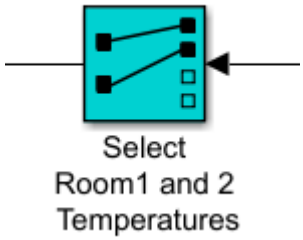


Рис. 2.7 Блок вибору температур приміщень з вектора температур (2 температури приміщень та 2 температури нагрівачів)

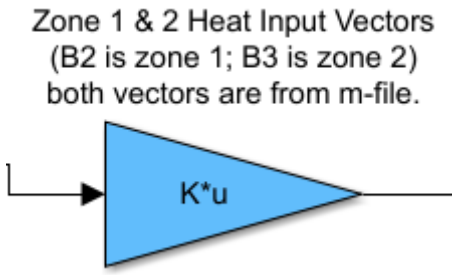


Рис. 2.8 Блок векторів теплових потоків в приміщеннях

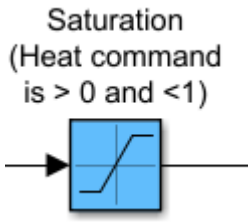


Рис. 2.9 Блок масштабування сигналу керування (від 0 до 1)

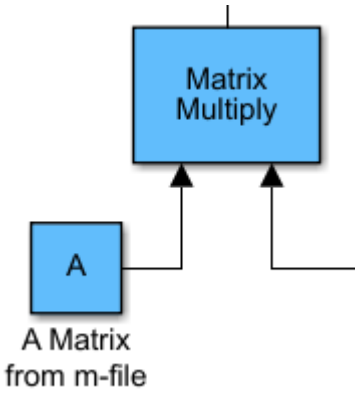


Рис. 2.10 Блок множення матриці об’єкту в просторі станів на вектор поточних температур приміщення та обігрівачів

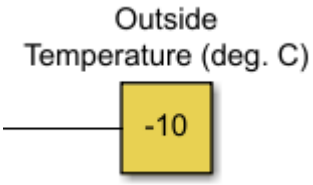


Рис. 2.11 Блок задання зовнішньої температури (стале значення)

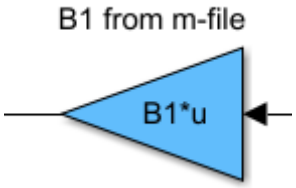


Рис. 2.12 Блок вектору теплових втрат до зовнішнього середовища

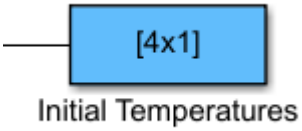


Рис. 2.13 Блок задання початкових температур в приміщенні та обігрівачів

Налаштування PID регулятора були визначені за допомогою PID Tuner. На рис. 2.14 та 2.15 наведено отримані налаштування регулятора та отриманий перехідний процес.

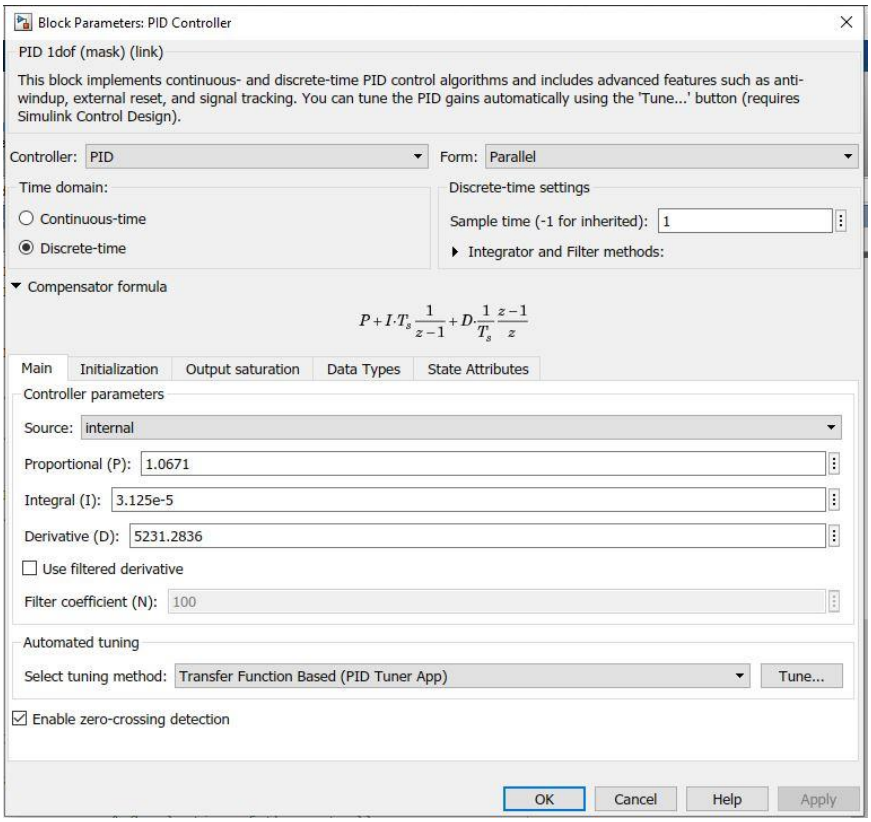


Рис. 2.14 Налаштування регулятора отримані в PID Tuner

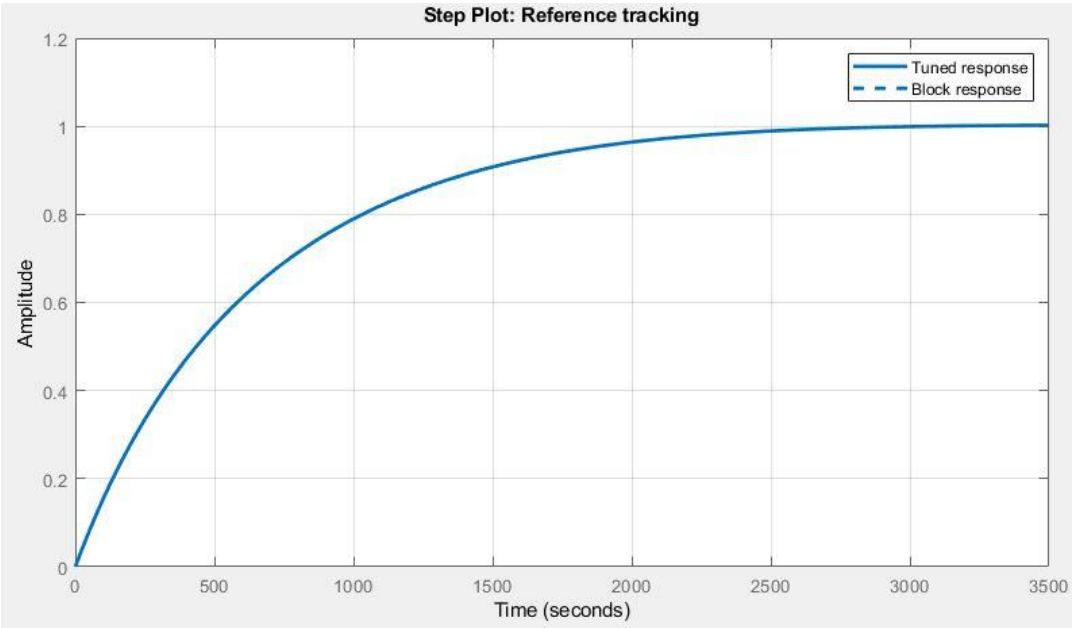


Рис. 2.15 Перехідний процес в PID Tuner

Для того щоб переконатись що система працездатна, виконаємо моделювання з виключеним обігрівом, включеним постійно на повну потужність

та включеним обігрівом та регулюванням за допомгою ПІД-регулятора. Результати моделювання представлено на рисунках 2.16, 2.17, 2.18.

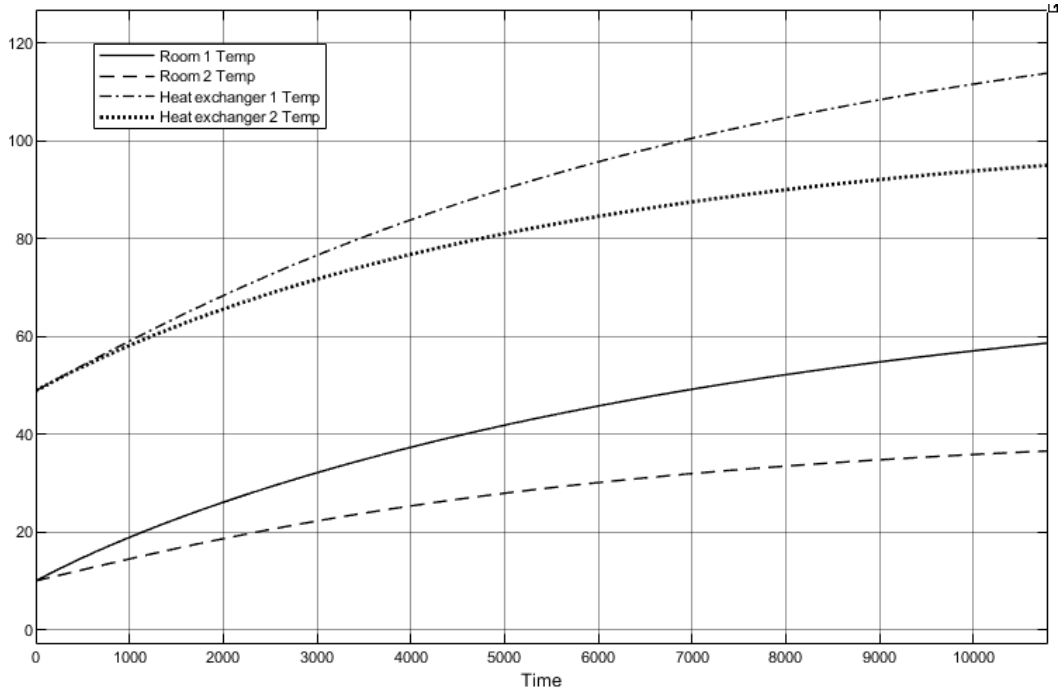


Рис. 2.16 Результат симуляції при без перервно включеному обігріві

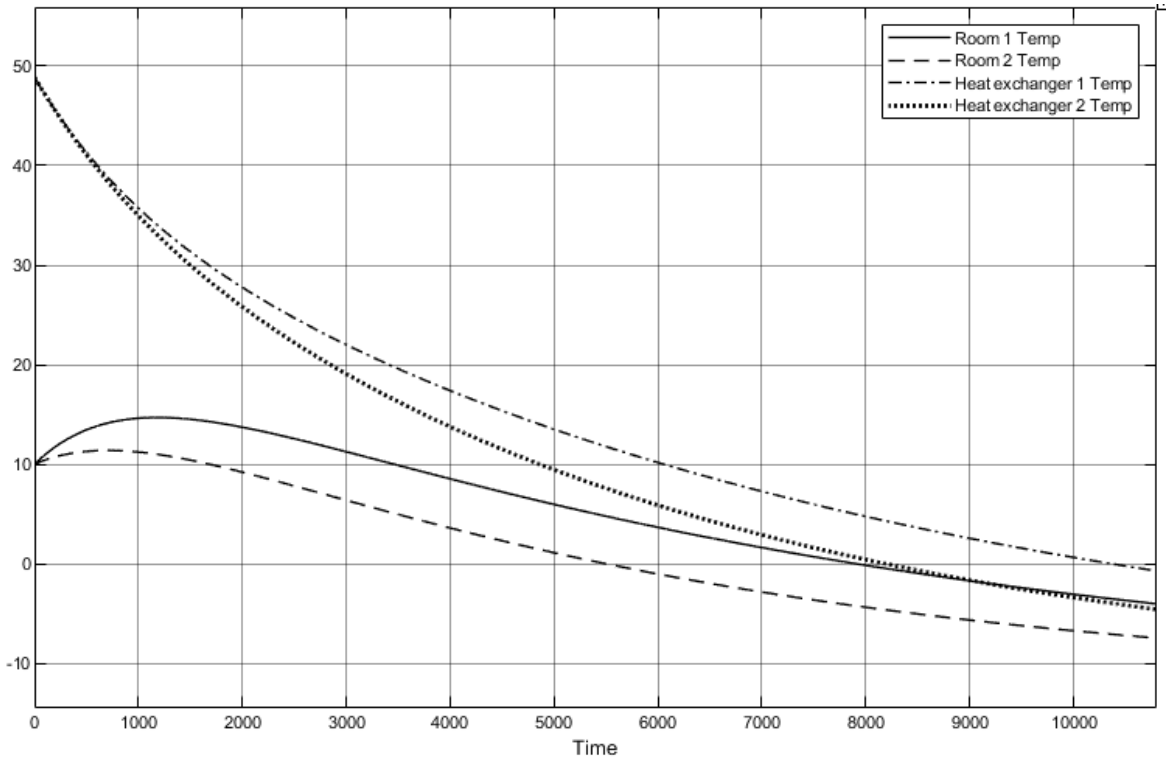


Рис. 2.17 Результат моделювання при виключеному обігріві

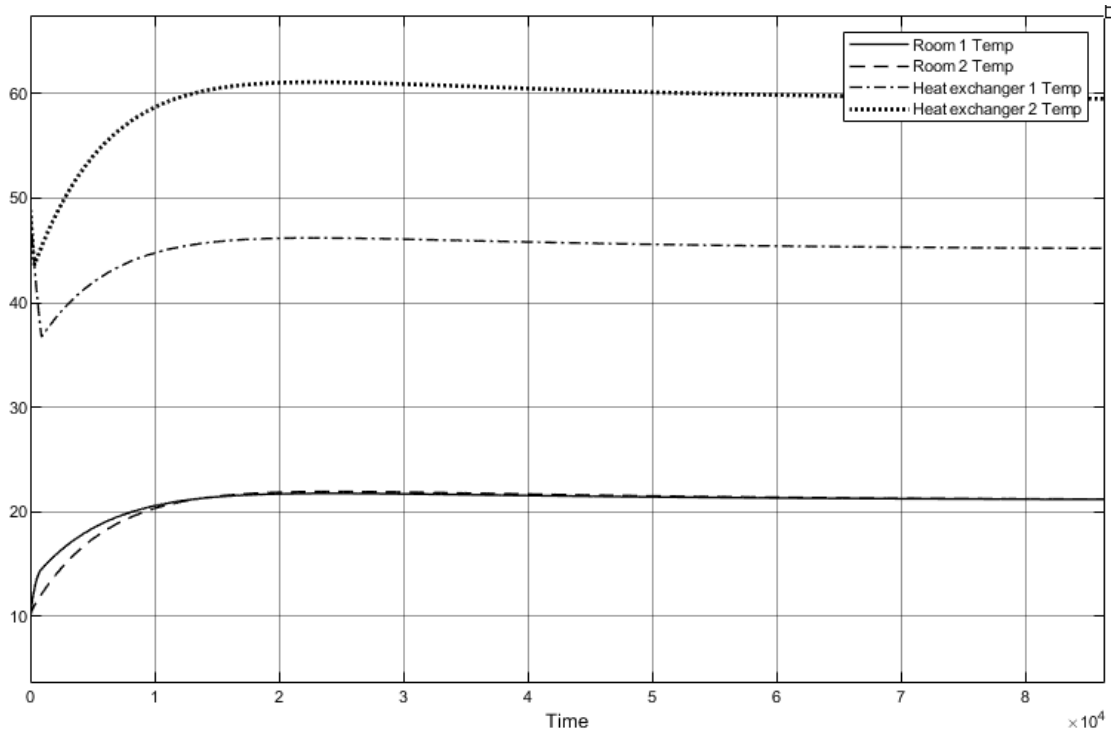


Рис. 2.18 Результат моделювання з використанням ПІД регулятора для отримання постійної температури (21 ºC)

Наступним кроком у процесі створення системи регулювання буде додавання до цієї моделі логіки регулювання. Це буде зроблено з використанням інструменту, який дозволяє реалізувати складну логіку з графічним потоком сигналу, подібним до Simulink (тобто із середовищем візуального програмування, "налаштованим" на потреби логіки). Таким інструментом є Stateflow, який є предметом наступного розділу. Ми переглянемо дану модель і створимо регулятор для обігріву будинку, який забезпечить більш комфотний обігрів. Система буде використовувати PID регулятор, який був використаний в даному розділі.

РОЗДІЛ 3
Розроблення системи управління об’єктом

3.1 Створення контролера для опалення будинку з використанням бібліотеки Stateflow

Першим кроком у розробці є створення моделі Simulink для фізичної системи, в якій буде працювати контролер. Ця частина процесу проектування є критичною, тому що вона дає основу для збору специфікацій, дозволяючи розробнику бачити вплив зміни вимог або специфікацій на спосіб роботи системи.

3.1.1 Створення моделі системи керування

Будемо вважати, що буде створено пристрій, який буде керувати системою опалення, яку було досліджено в попередньому розділі. Розділимо систему на чотири частини:

- процес горіння,
- теплообмінник, який перетворює газ або нафту в гарячу воду
- насоси, які подають нагріту воду до теплообмінників у приміщеннях,
- динаміка теплообміну в приміщеннях (надходження тепла в приміщення від теплообмінників і втрата тепла через провідність, проникнення зовнішнього повітря та випромінювання).

Ми змоделювали будинок, який має дві "кімнати" в попередньому розділі. Ця модель мала всі атрибути багатокімнатного будинку, але дану модель було легше побудувати. Початкове моделювання має тенденцію бути простим, але розширюваним. Коли системні інженери говорять про поточну фазу, вони часто

говорять про "скорочення" вимог вниз. Ця фраза фіксує один з найважливіших атрибутів розвитку специфікації, що починається з вимоги до системи як повноцінної одиниці. Визначається сама верхня частина системи; в цьому випадку потрібно енергоефективна система опалення, яка забезпечує контроль над кожним окремим приміщенням у багатокімнатному будинку з теплом, що надходить у приміщення постійно, не включається та вимикається терморегулятором. Наступним кроком є моделювання динаміки для дослідження взаємодії різних частин системи. Саме тут проста модель відіграє значну роль. Так звані компроміси зараз можуть мати місце, коли більш складна підсистема в одній області може ускладнити проектування інших підсистем. Компроміс більш чітких специфікацій в одній області для менш обтяжливих специфікацій в іншій вимагає дуже надійної моделі. Насправді це єдиний метод, який розробник може використовувати для розробки складної специфікації. Однак у більшості інженерів рання робота зі створення цих моделей відсутня, оскільки письмова специфікація - це все, що бачать інженери на наступних етапах проектування. Передача специфікації через письмовий документ, який виключає симуляції, що використовуються для її розробки, призводить до втрати інтелектуального контенту, і це значно уповільнює процес проектування.

Для того, щоб зрозуміти процес, буде створено новий регулятор опалення будинку, використовуючи модель будинку, що була розроблена в попередньому розділі. Імовірно, модель стала відправною точкою для проекту нового регулятора тепла, і модель показала межі про те, що можливо за допомогою нового контролера. Підсистема Simulink, яка містить контролер, зображена на рисунку 3.1. Характерні особливості цієї підсистеми такі:

- система використовує PID-контролер, що має сенс, оскільки система опалення повинна забезпечувати власнику будинку постійну

температуру за будь-яких умов. (Інтегральна складова змушує температуру відповідати заданій.)

- диференційна складова дозволяє контролеру передбачати зміни зовнішніх температур та підлаштовуватись під них.
- пропорційна складова змушує опалювальну систему дотримуватися змін температури в приміщенні, викликаних порушеннями, такими як відкривання та закривання дверей.
- система управління є цифровою. (Це неявно в специфікації, оскільки використано цифровий пристрій для управління; час вибірки- 1 сек., що є більш ніж адекватним, але достатньо довгим, щоб не викликати жодних обчислювальних проблем.)

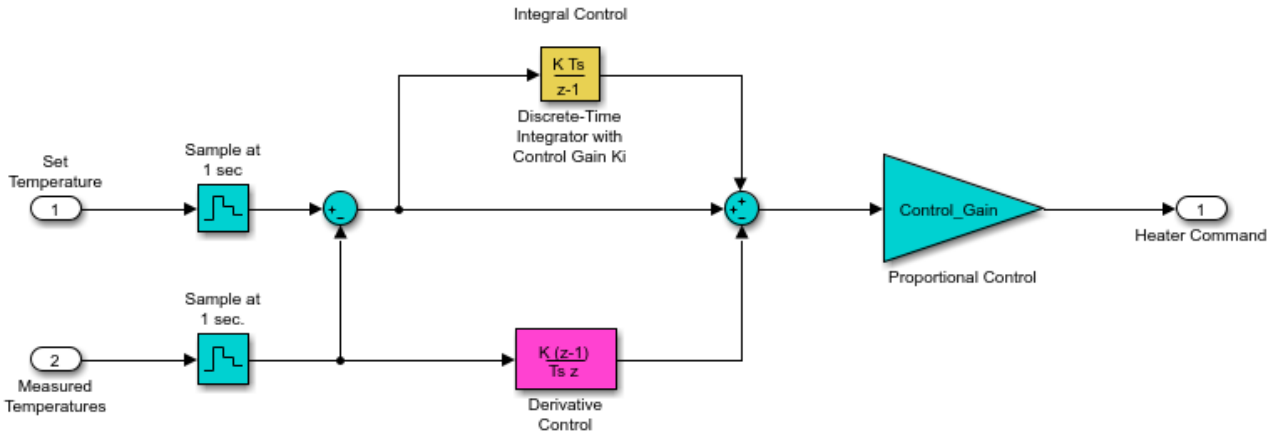


Рис. 3.1 Модель ПІД контролера для системи опалення будинку

Результати моделювання за допомогою специфікації, що виконується, хороші, але ми, повинні бачити, що відбувається, коли зовнішня температура змінюється. Підготований температурний профіль (з реальних даних) для типового дня в Києві буде використано для коливанн температури. Для цього додамо блок, що зчитує дані з робочої області MATLAB – Workspace, попередньо загрузивши дані в матричну змінну з excel файлу. Модель системи з блоком що зчитує підготовлені дані про погоду з Workspace MATLAB показано на рис 3.2.

Графік зміни зовнішньої температури та температури в приміщенні протягом доби показано на рис. 3.3.

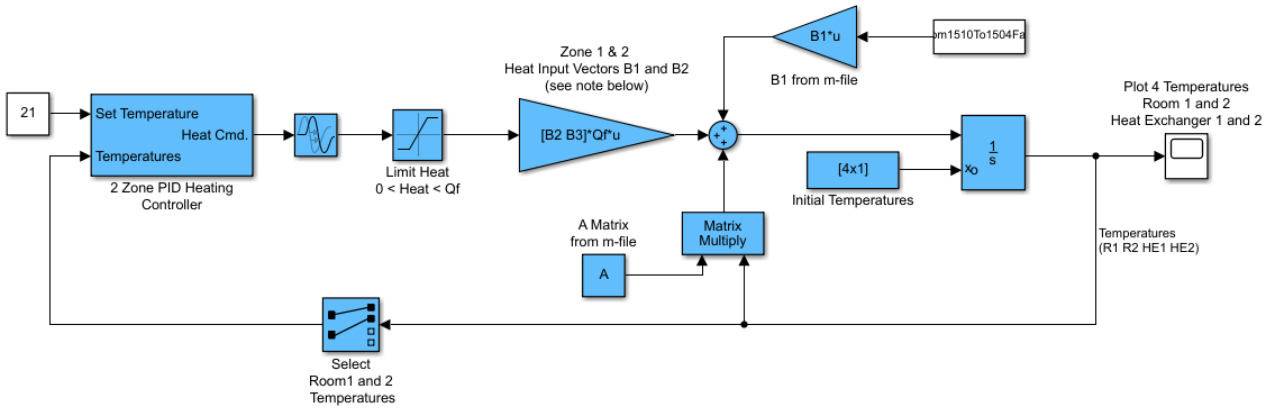


Рис. 3.2 Модель системи з підготовленими даними про погоду

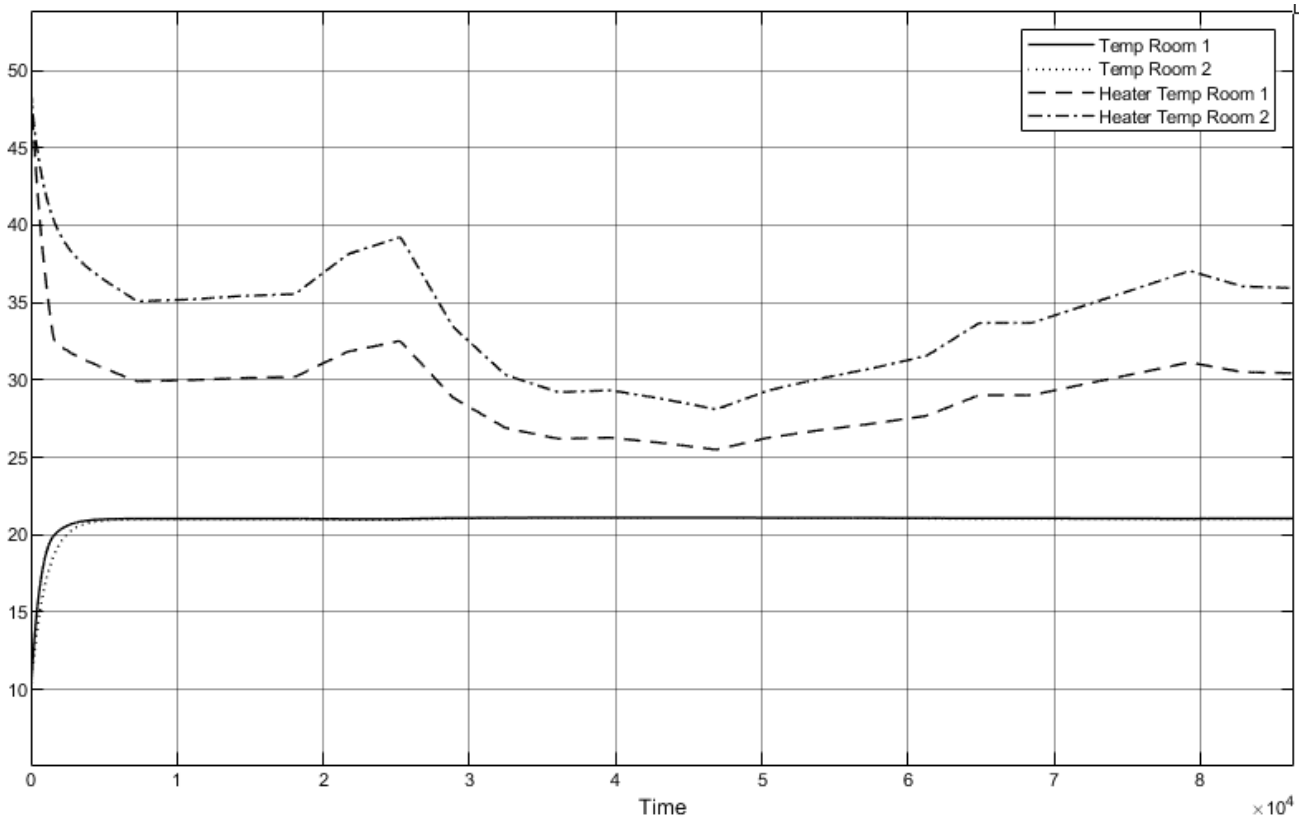


Рис. 3.3 Робота системи керування при зміні зовнішньої температури

Тепер зовнішня температура в моделі слідує профілю зовнішньої температури яку було підготовлено раніше. Зауважте, що температура в приміщенні змінюється від 10 ° C до встановленої точки 21 ° C, приблизно за 20

хвилин. Крім того, контролер опалювальної системи підтримує постійну температуру в 21 градусів С для обох приміщень, незважаючи на добові коливання температури. Крім того, температура циркулюючої води піднімається вгору і вниз, коли змінюється зовнішня температура; це допомагає підтримувати постійну температуру стін і значно покращує рівень комфорту в приміщеннях. Це моделювання дозволяє переконатися, що цей контролер працює добре. На даний момент є все необхідне для створення блоку Stateflow, який реалізуватиме цей контролер.

Майбутній контролер повинен мати можливість підвищити та знизити задану температуру з пристрою, відображати задані температури для кожної зони, відображати стан системи для користувача, включаючи поточний час та поточну температуру в приміщенні для кожної кімнати (зони). Припустимо, що буде реалізовано орієнтовний інтерфейс на рисунку 3.4 для регулятора опалення. З цього сформовані вимоги до логіки яку потрібно реалізувати в Stateflow.

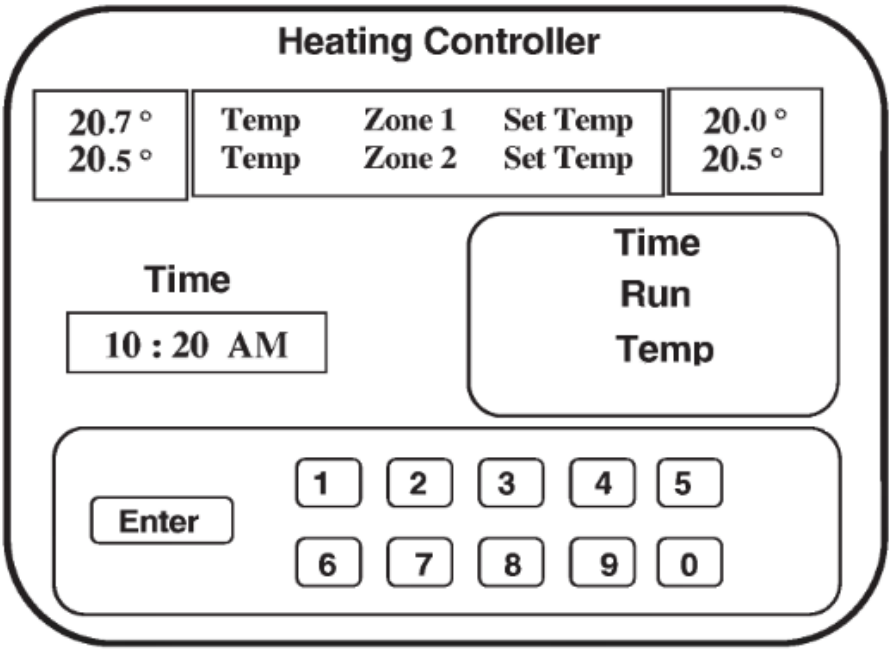


Рис 3.4 Орієнтовний інтерфейс контролера для опалення будинку

3.1.2 Компоненти теплового регулятора

Перший крок розробки - написати специфікацію для діаграми Stateflow, яка окреслює всі дії, які може виконувати користувач. Дії будуть наступні:
Користувач може вибрати будь-який з трьох режимів роботи контролера: Run, Set Time та Temp.

- В режимі «Run» контролер працює для підтримки встановленої температури в кожній з зон. Контролер також відображає час доби.
- В режимі «Time» користувач може встановити час. Користувач встановлює час за допомогою кнопкової клавіатури (з цілими числами від 1 до 0) та кнопкою Enter. Натискання клавіші Enter зберігає час на дисплеї. Зображення переходить зі значення годин і хвилин кожного разу при натисканні нової цифри. Якщо перед введенням повного значення часу натиснути клавішу Enter, введення розпочнеться повторно.
- В режимі «Temp» користувач може встановити температуру для кожної зони. Відображення починається з зони 1. Бажану температуру встановлюють за допомогою цифрових клавіш. Після введення кожної цифри можна змінити її (натискання нової цифри змушує дисплей переходити через дві цифри). Натиснувши клавішу Enter, введене значення зберігається.
- Зони вибираються за допомогою клавіші введення відразу після вибору параметра «Встановити темп».
- У всіх випадках система продовжує працювати належним чином, навіть якщо перемикач не перебуває у положенні "Запуск".

3.1.3 Додавання користувацьких дій та Stateflow діаграми до Simulink моделі

Модель Simulink потребує симуляції різних дій користувачів, щоб можна було протестувати різні режими роботи діаграми Stateflow. Під час тестування діаграми, необхідно переконатися, що охоплено всі можливі дії і що кожен стан в діаграмі використовується. Нам також потрібне моделювання цифрового годинника, який запустить контролер. Цифровий годинник також використовується для відображення часу на контролері. Графічний інтерфейс був створений за допомогою GUIDE (GUI Development Environment) в MATLAB. GUI імітує всі взаємодії користувачів, у тому числі

- відображення часу,
- відображення фактичної температури в приміщенні,
- відображення заданої кімнатної температури,
- режими роботи: "Run", "Set temp" "Set time",
- функціональна клавіатура, яка дозволяє користувачеві вводити значення;

Коли користувач змінює налаштування, вони одразу стають доступними в моделюванні, і графічний інтерфейс відображає результат.

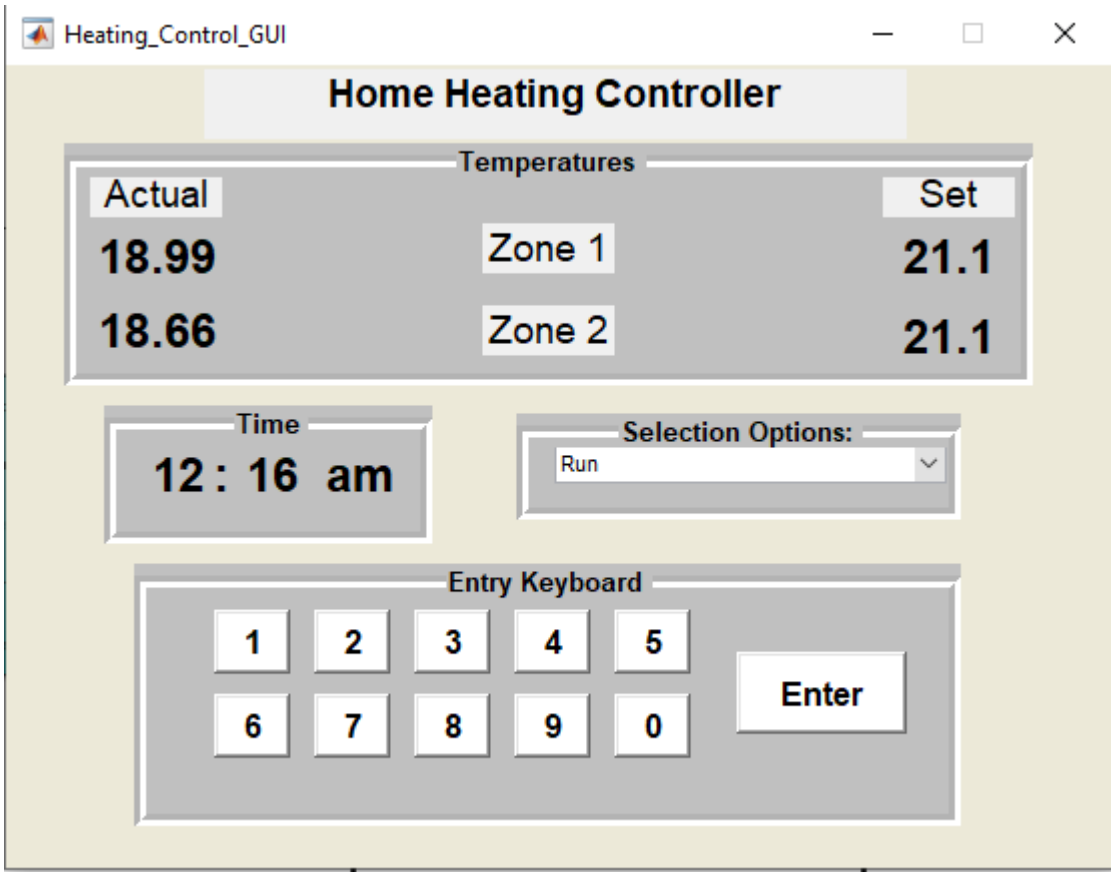


Рис. 3.5 Графічний користувацький інтерфейс контролера

Далі буде розглянуто всі деталі створення цього графічного інтерфейсу, але спочатку описано як він працює. Модель Simulink з контролером та всім кодом для керування графічним інтерфейсом наведені нижче. Діаграма Stateflow включає логіку для управління зворотним зв'язком (команда контролеру опалення надходить з порту Command в діаграмі) і всіх параметрів кнопки та вибору. До них відносяться параметри роботи системи та зміни часу та бажаних температурних параметрів.

Модель включає цифровий годинник, який керує дисплеєм та запускає Stateflow діаграму, блок Clock. П'ять блоків MATLAB обробляють зміни GUI:

- Оновлення часу на дисплеї.

- Скидання часу в графічному інтерфейсі: Коли користувач змінює час, вибравши «Set Time» у випадаючому меню, цей код змінює відображення часу в графічному інтерфейсі на час, введений користувачем.
- Кнопки скидання в блоці "User_selections": зв'язок між графічним інтерфейсом та моделлю Simulink реалізовується через блок Gain, значення якого змінюється кодом у M-файлі GUI.

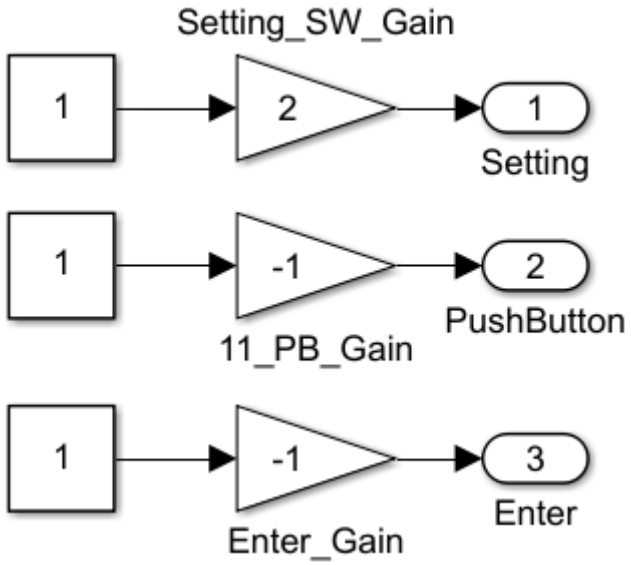


Рис. 3.6 Блок User_Selections для взаємодії між користувацьким інтерфейсом та діаграмою Stateflow

- Оновлення заданих температур в GUI: користувач вводить задані температури з клавіатури, цей код відображає їх у графічному інтерфейсі.
- Оновлення поточних температур в кімнатах в GUI: вимірювані кімнатні температури є результатом блоку Simulink, який імітує перетворення A / D вимірюваних температур на 3 цифри (у формі XX.X), які відображаються в графічному інтерфейсі.

Цифровий контролер та модель приміщень точно такі ж як в моделі розробленій в попередньому розділі. Додано порожню діаграму Stateflow до моделі, та додали наступні входи в діаграму:

- Режим роботи (ціле число від 1 до 3 у випадаючому меню GUI);
- 11 значень з нажатих кнопок, які походять від взаємодії GUI та блоку User_Selections (ці значення —1, 0, 1, 2 ..... 9);
- результат натискання кнопки Enter (при скиданні кнопок значення становить –1, і воно змінюється на +1 при натисканні кнопки Enter; що є результатом взаємодії графічного інтерфейсу з блоком User_Selections).

Вхідні дані блоку User_Selections (зліва від моделі) є результатом зміни в коефіцієнтах підсилення в блоках Gain. Ці блоки Gain знаходяться у блоці User_Selections, показаному на рисунку 3.7.

Зміна значення відбувається за допомогою взаємодії Simulink з MATLAB, команда в MATLAB - set_param, і вона дозволяє змінювати значення параметрів параметрів у блоці. Для блоку Gain - це лише два параметри - сам коефіцієнт посилення та час вибірки. Для того, щоб використовувати setparam, потрібно отримати ключ до графічного об’єкта (блоку), який параметри якого треба змінити з графічного інтерфейсу. Приклад виконання даної операції:

```
set_gain = ‘1’;
block_handle =
‘Stateflow_Heating_Controller/User_Selections/Setting_SW_Gain’;
set_param(block_handle, ‘Gain’, set_gain);
```

За таким же принципом працюють всі інші кнопки графічного інтерфейсу.

На рис. 3.8 зображено Simulink блок діаграми Stateflow, зі всіма входами та виходами.

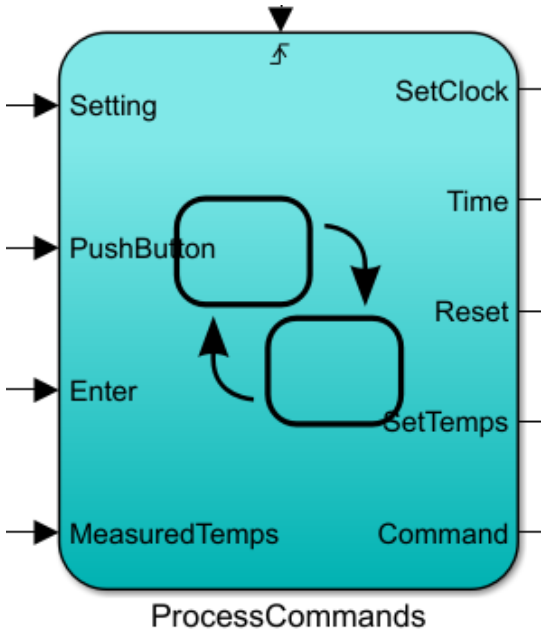


Рис. 3.8 Блок Stateflow діаграми в Simulink

Діаграма Stateflow має два паралельних стани, які називаються Interactions (рис. 3.9) та Process_Buttons (рис. 3.10).Стан Interactions обчислює дві команди для контролера (кожна команда - різниця між фактичною та заданою температурою для кожної зони) та обробляє зміни стану перемикача. Другий паралельний стан обробляє натискання кнопок.

Дані діаграми досить прості для розуміння. Вибір одного з трьох режимів роботи "Run", "Time" або "Temp" призводить до того, що діаграма залишиться в такому стані . У стані SetTime очікується нажаття кнопки. Коли відбувається нажаття, значення змінної "PushButton" дорівнює 0, 1, … 9, залежно від того, яка кнопка була натиснута, діаграма переходить до стану Hours2, і запамятовує значення введеної першої цифри годин. Далі введене значення з кнопки скидається і діаграма переходить в стан очікування нажаття кнопки, якщо за наступних 45 секунд значення не було введено, то режим введення годин вимикається і всю процедуру потрібно буде повторити заново. Якщо ж користувач ввів наступну цифру за 45 секунд очікування, то діаграма запамятовує другу цифру в значенні годин та знову переходить в режим

очікування введення і так далі поки не буде введено години в форматі hh:mm та нажато кнопки enter для встановлення заданого значення. Процес задання значення температури починається з блоку Temperature. При виборі режиму роботи “Set temp”, діаграма очікує нажаття кнопки Enter, при кожній такій події змінюється поточна зона для встановлення температури. Коли потрібна зона вибрана задається значення температури за тим же принципом як і задається значення часу в режимі “Set time”.

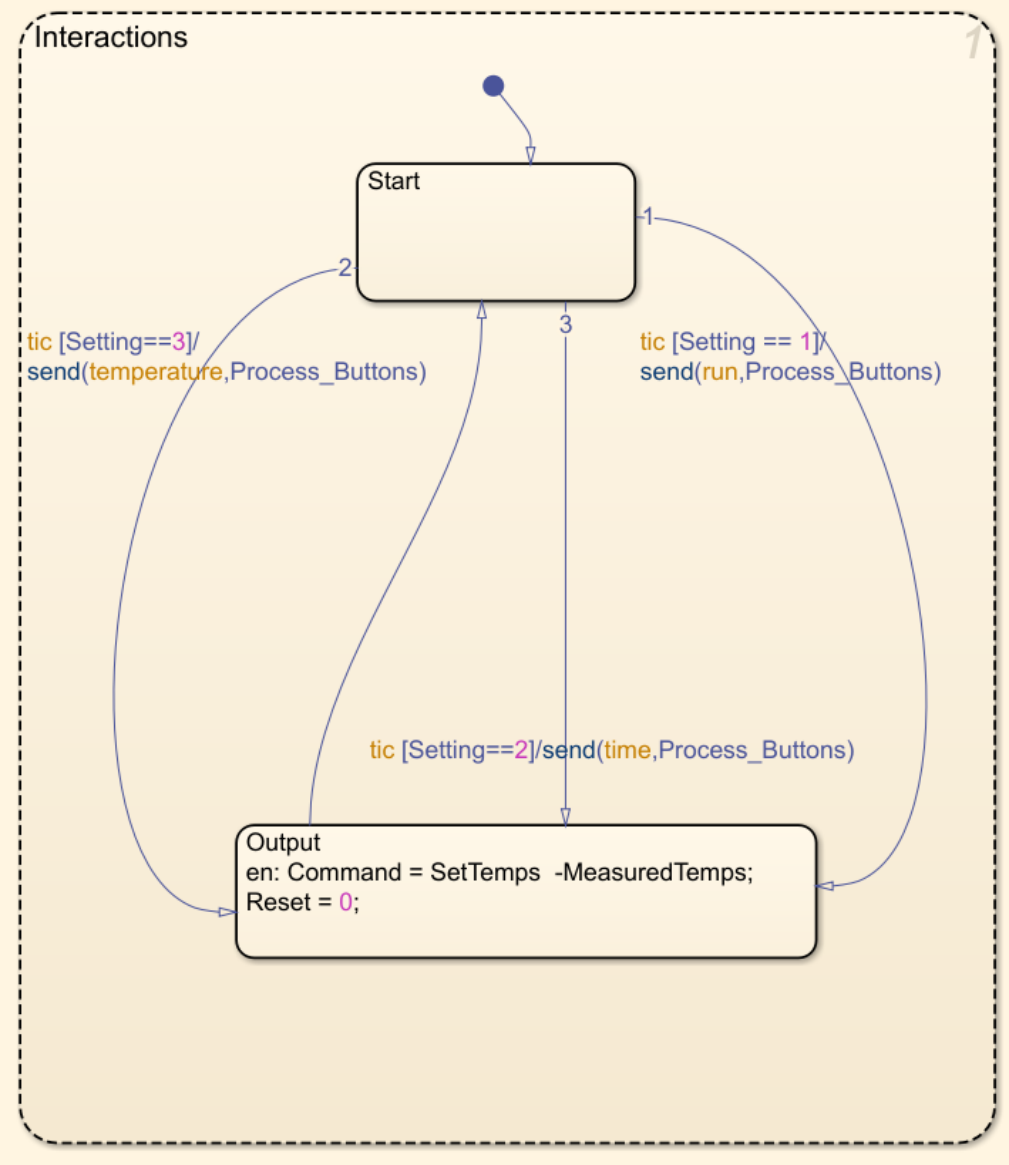


Рис. 3.9 Діаграма Stateflow стану Interactions

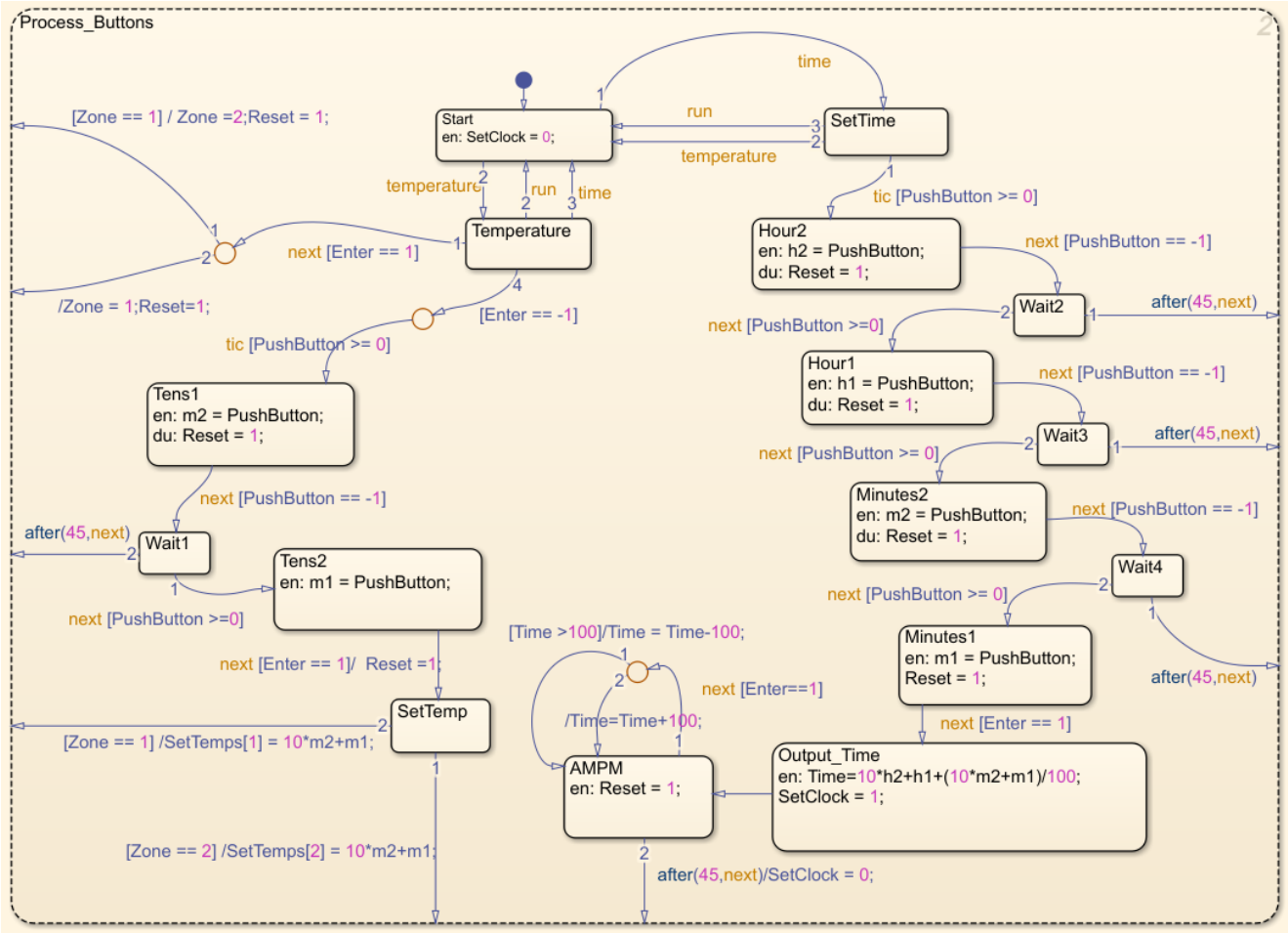


Рис. 3.10 Діаграма Stateflow стану Process_Buttons

Також в моделі використано блок перетворення аналогового сигналу в дискретний. Даний блок в Simulink та його структура зображено на рис. 3.11 та 3.12 відповідно.

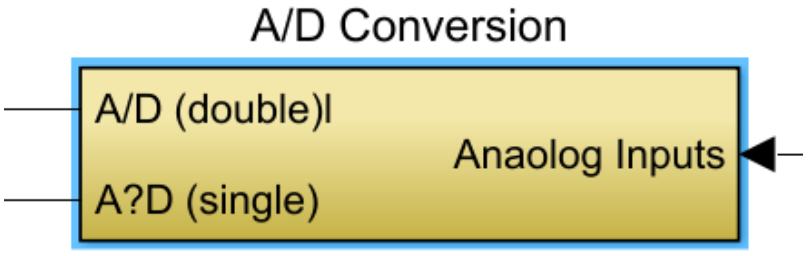


Рис. 3.11 Блок А/Д перетворення

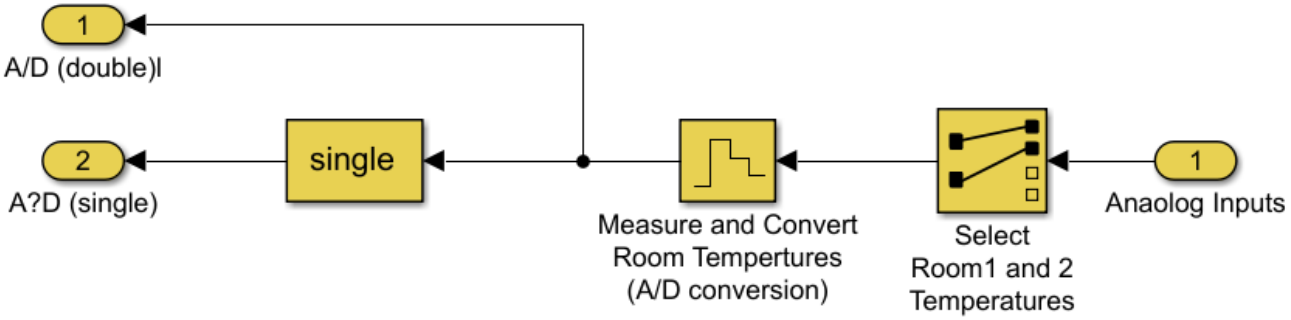


Рис. 3.12 Структура блоку А/Д перетворення

Єдиними частинами моделі які залишилися є блоки коду MATLAB, які змінюють відображенні в GUI. Один з блоків - код, який скидає значення кнопки, вже було наведено раніше. Інші чотири усі виконують зміни в GUI. Дві з них вносять зміни, якими користувач командує за допомогою кнопок, а інші два змінюють час та температуру в міру зміни стану діаграми. Далі для прикладу розглянуто роботу блоку що задає час в GUI.

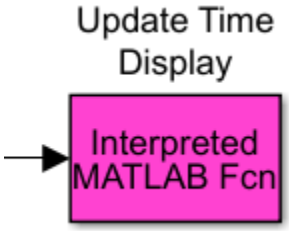


Рис. 3.13 Блок виклику функції MATLAB updateTime для оновлення часу в GUI

Код блоку:

```
function updatetime(Clock)
persistent AMPM Clocksec Clockmin Clockhr changehr changemin changeampm

if Clock == 0; return; end
hGUI     = evalin('base','hGUI');
h  = guidata(hGUI);
Hrfield  = get(h.text15,'String');

if strcmp(Hrfield,'00')
   AMPM      = 'am';
   Clocksec  = 0;
   Clockmin  = 0;
   Clockhr   = 12;
   changemin = 1;
```

```
    changehr  = 1;
    changeampm= 1;
end

Clocksec = Clocksec + 1;
if Clocksec == 60
    Cm = get(h.text13,'String');
    Clockmin = str2double(Cm);
    Clocksec = 0;
    Clockmin = Clockmin + 1;
    changemin = 1;
    if Clockmin == 60
        Ch = get(h.text15,'String');
        Clockhr   = str2double(Ch);
        Clockhr   = Clockhr + 1;
        Clockmin  = 0;
        changehr  = 1;
        if Clockhr == 13; Clockhr = 1; end
        if Clockhr == 12 && Clockmin == 0
            changeampm = 1;
            if strcmp(AMPM,'am')
                AMPM = 'pm';
            else
                AMPM = 'am';
            end
        end
    end
end


if changemin == 1;
    Clockmins = num2str(Clockmin);
    if Clockmin < 10; Clockmins = ['0' Clockmins]; end
    set(h.text13,'String',Clockmins)
    changemin = 0;
end
if changehr == 1;
    Clockhrs = num2str(Clockhr);
    set(h.text15,'String',Clockhrs)
    changehr = 0;
end
if changeampm == 1;
    set(h.text11,'String',AMPM)
    changeampm = 0;
end
```

Даний програмний код знаходить об’єкт у графічному інтерфейсі та змінює його атрибут, у даному випадку змінює текст елементу що відображає час. Для

того щоб знайти об’єкт було використано функцію evalin що використовується для отримання об’єкту по ключу який є властивістю об’єкта.

За такми же принципом працюють блоки для скидання часу (resetTime), оновлення заданих температур (updateSetTemps), оновлення температур в приміщеннях (updateRoomTemps). Дані блоки зображено на рис. 3.14.

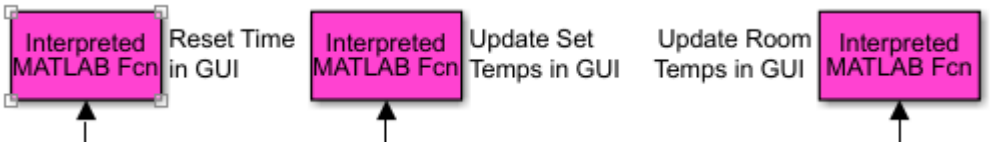


Рис. 3.14 Блоки виклику функцій resetTime, updateSetTemps, updateRoomTemps

Контролер (з ПІД керуванням) та діаграма Stateflow (разом з перетворювачами A / D) - це єдині блокі цієї моделі, які будуть використовуватися у фактичному керуванні. На рис. 3.15 показана фінальна версія блоку ПІД регулювання.

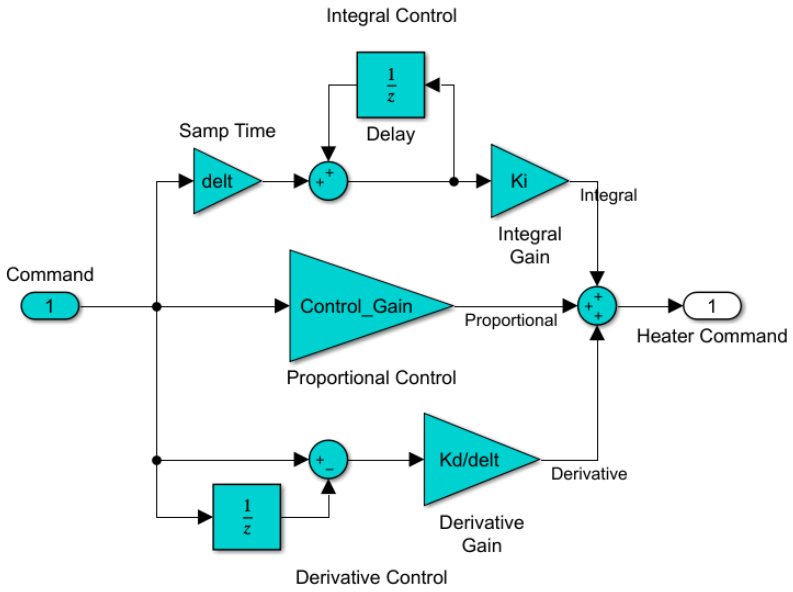


Рис. 3.15 Структура блоку ПІД регулювання

На рис. 3.16 зображено модель системи керування опаленням в будинку в Simulink з блоком діаграми Stateflow та обробкою дій користувача. На рис. 3.17 та 3.18 зображено приклад роботи Stateflow діаграми в повільному режимі, що

дозволяє від проводити відладку розробленої системи керування та наочно показує в якому стані система перебуває в даний час та якою віткою переходить в наступний стан. На рис. 3.19 зображено графік температур в будинку та температур обігрівачів зі зміною заданої температури та зміною зовнішньої температури.

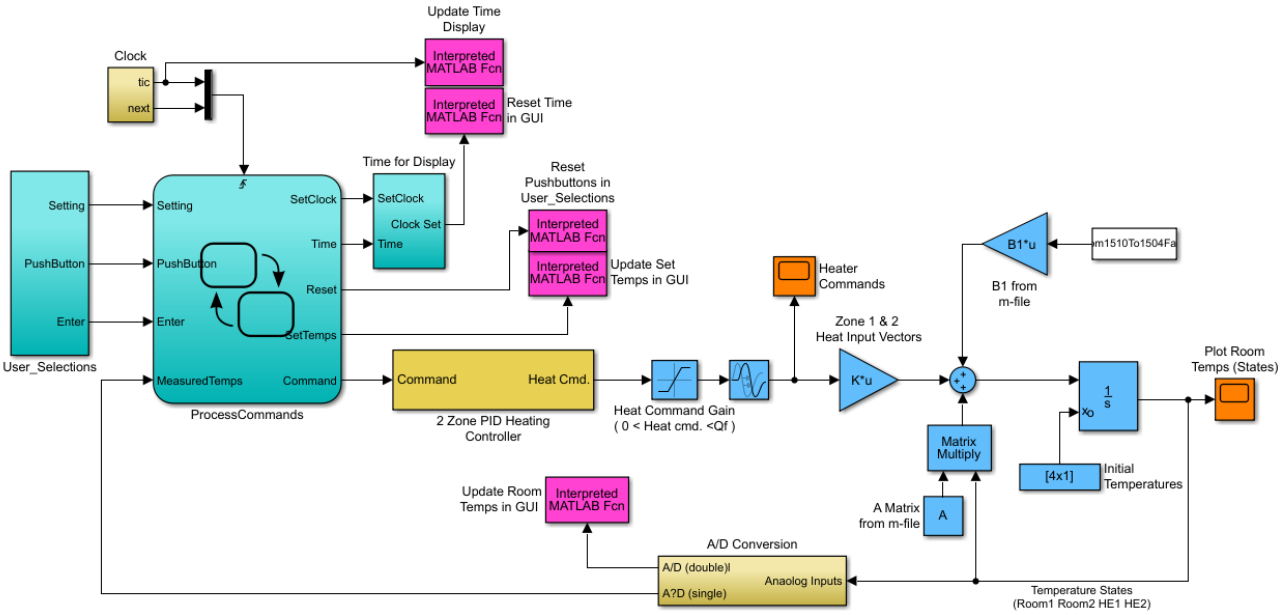


Рис. 3.16 Модель системи регулювання з доданим блоком діаграми Stateflow та обробкою дій користувача

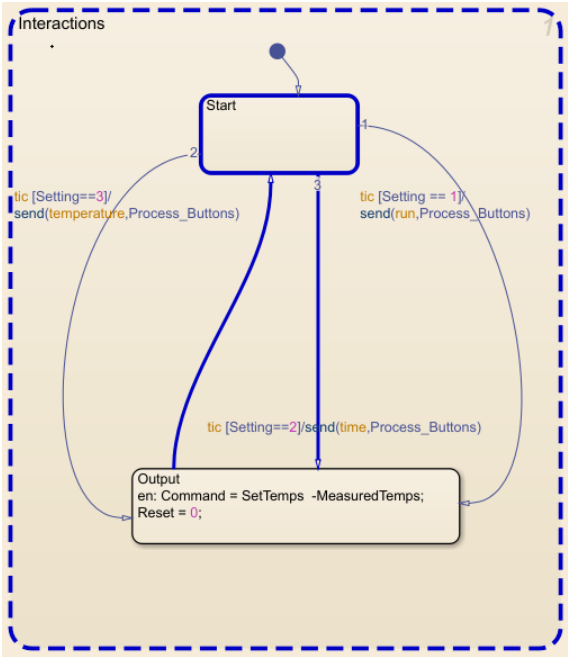


Рис. 3.17 Приклад роботи діаграми Stateflow (стан Interactions)

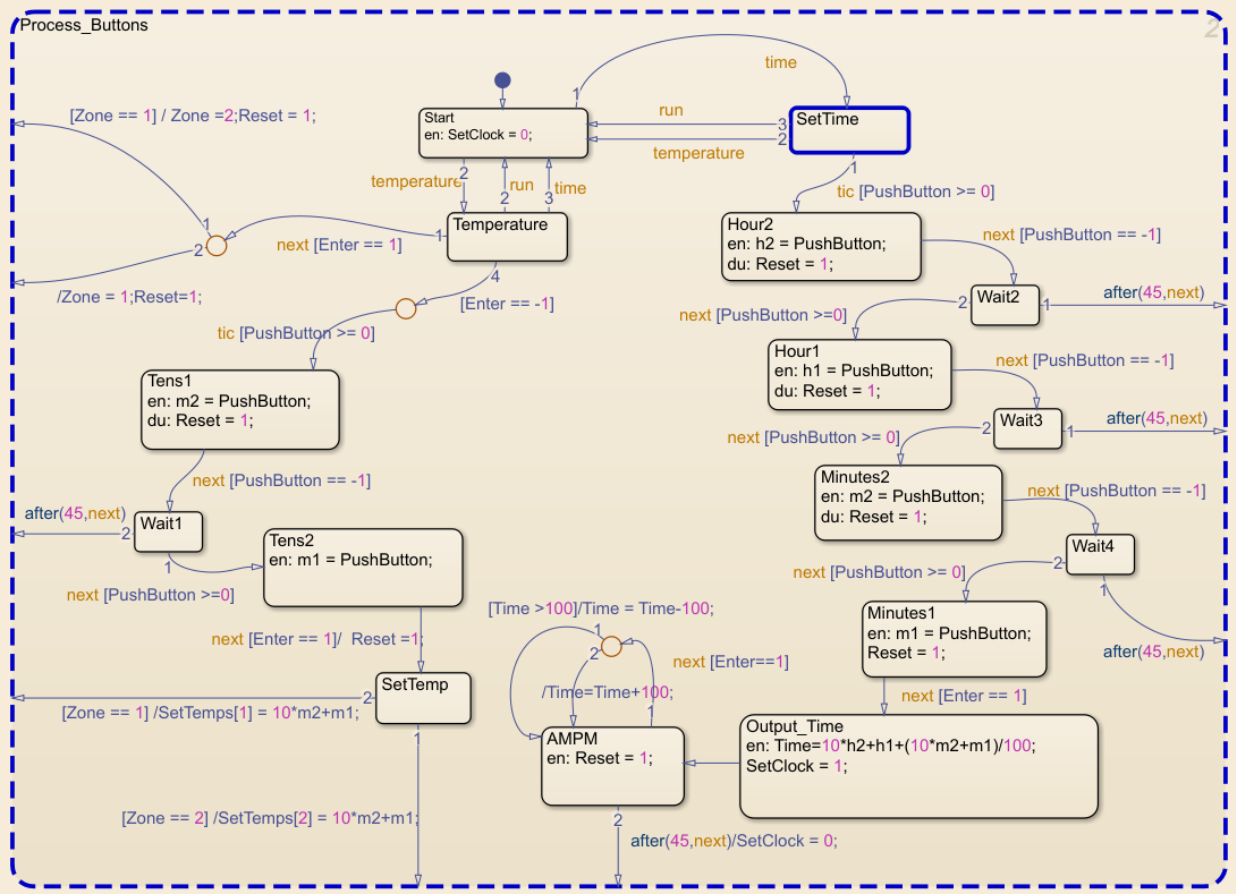


Рис. 3.18 Приклад роботи діаграми Stateflow (стан Process_Buttons)

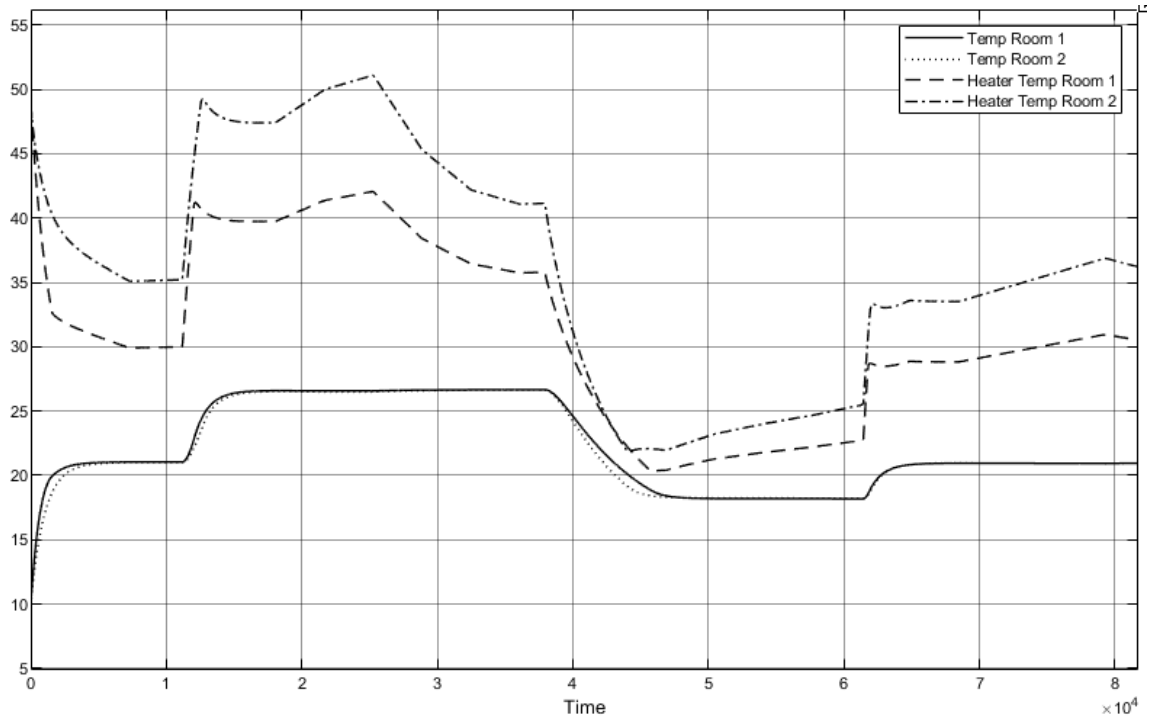


Рис. 3.19 Графік температур в будинку зі зміною заданої температури та температури зовнішнього середовища

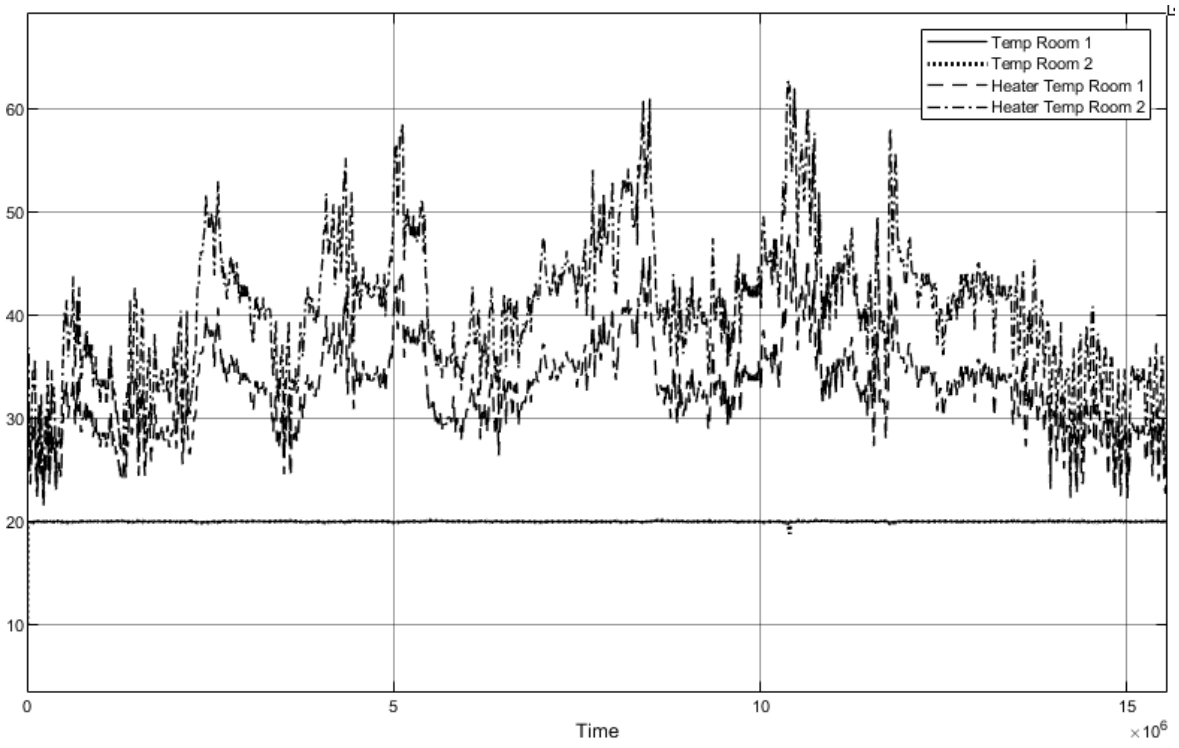


Рис. 3.20 Графік підтримки заданої температури в приміщенні протягом опалювального сезону

3.1.4 Генерація C/C++ коду зі Stateflow моделі для використання у контролері

Для створеної моделі Stateflow є можливість генерації C/C++ коду для використання у контролерах, що підтримують ці мови програмування. Щоб згенерувати код, достатньо вибрати блок Stateflow в Simulink, через налаштування якого вибрати пункт меню “Build this subsystem”. Надалі після першої генерації, код можна відкрити через пункт меню “Open subsystem report” На рис. 3.21 показано меню вибору вище описаних функцій. На рис. 3.22 показано вікно зі згенерованим кодом.

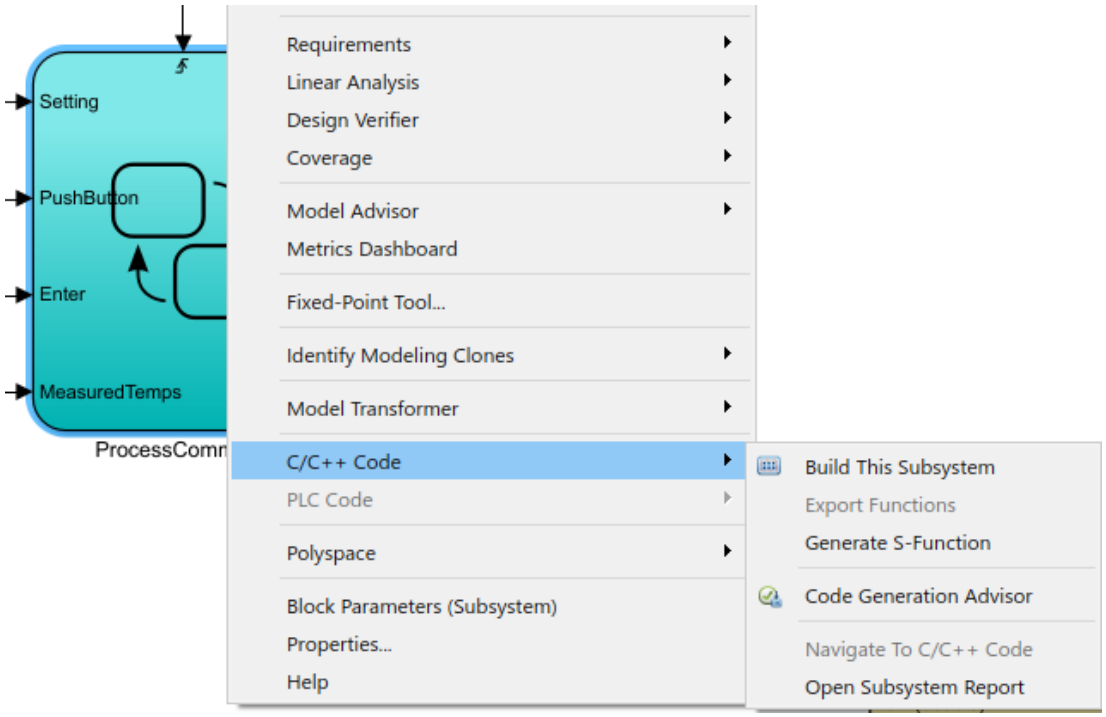


Рис. 3.21 Меню вибору генерації коду для блоку Stateflow

Згенерований код з діаграми Stateflow ProcessCommands, розробленої в даній роботі наведено в додатку А.

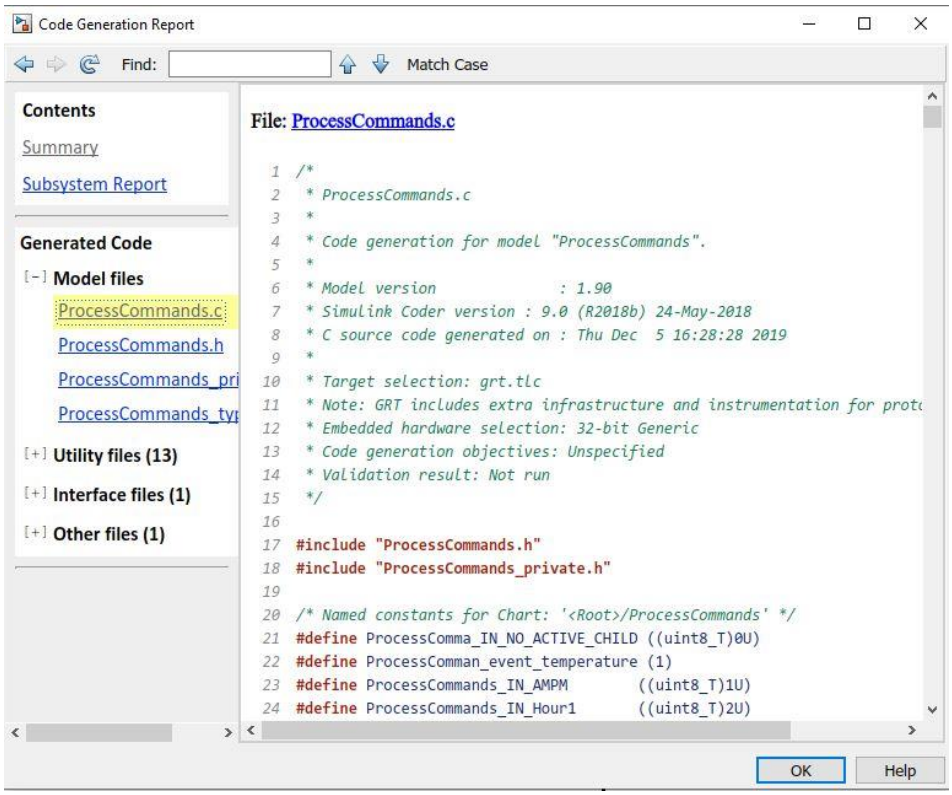


Рис. 3.22 Вікно відображення згенерованого коду

3.1.5 Порівняння ефективності PID регулятора та термостата

Для порівняння ефективності системи керування з PID регулятором та системи керування з термостатом було створено прості моделі, що складаються з моделі об’єкту (приміщення для обігріву), блоку регулювання та блоку зчитування метеоданих на весь опалювальний сезон. Під час моделювання задана температура в приміщенні була рівною 21 ºС. Зона нечутливості термостата була 2 ºС. Систему керування з PID регулятором представлено на рис. 3.23. Систему керування з термостатом представлено на рис. 3.24.

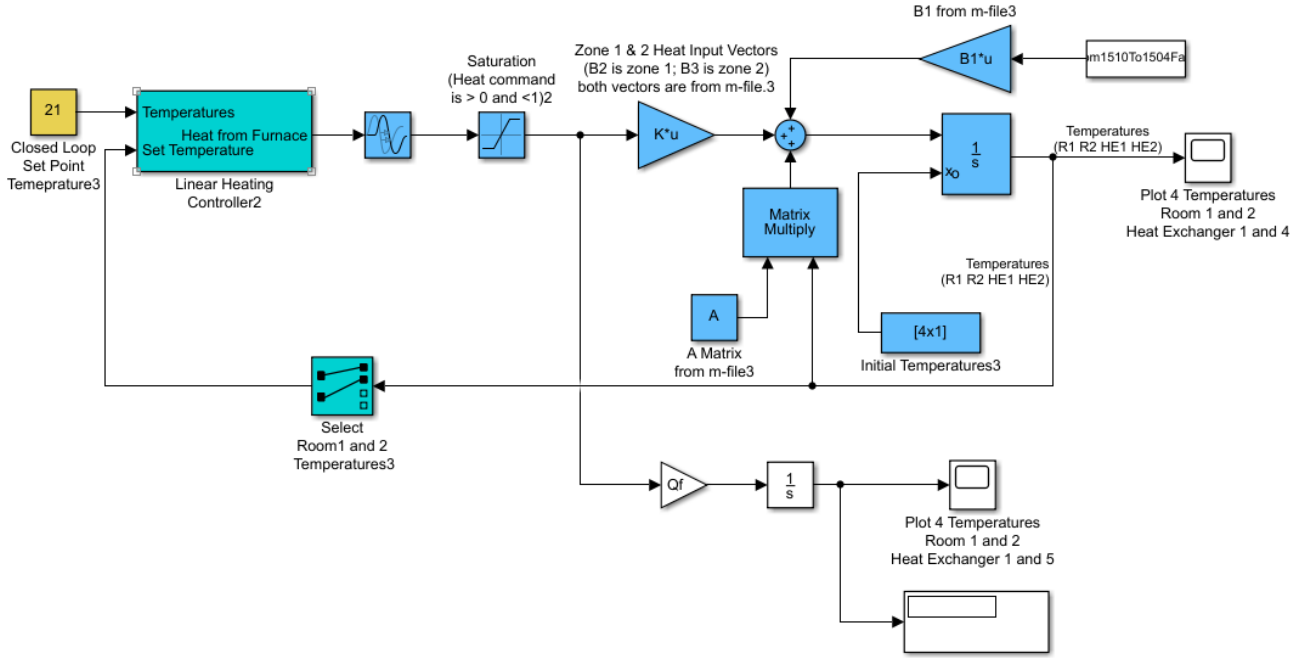


Рис. 3.23 Система керування з PID регулятором

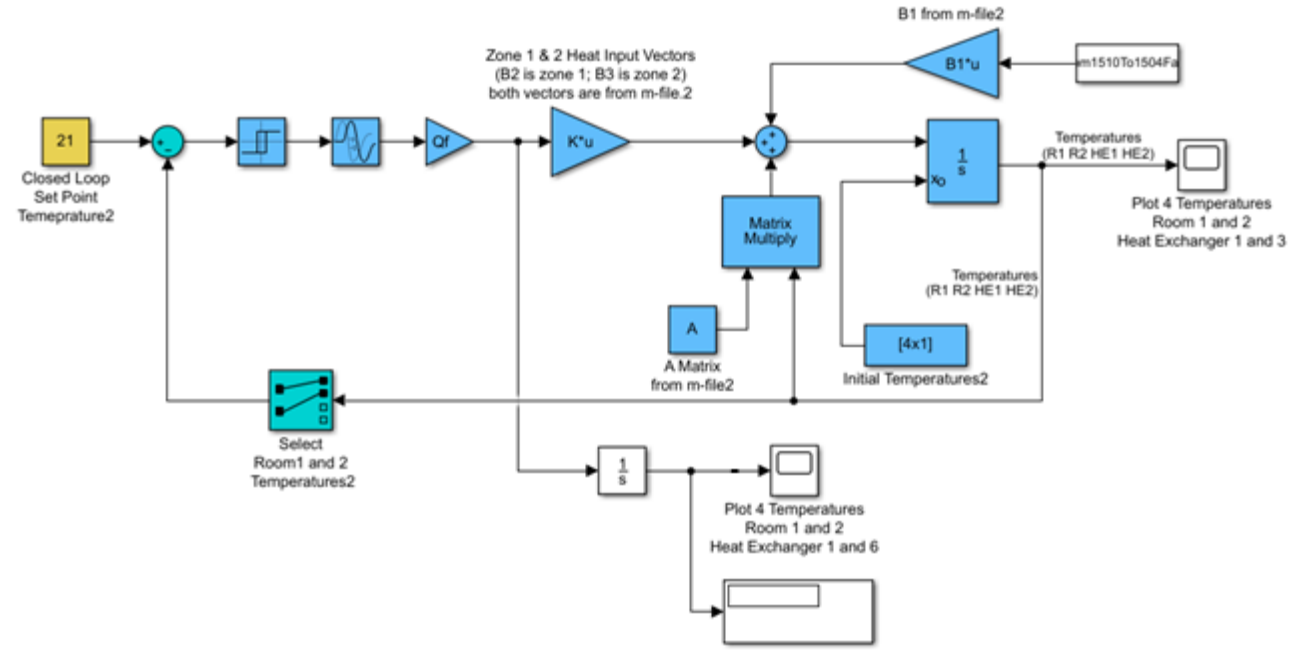


Рис. 3.24 Система керування з термостатом

За результатами моделювання PID регулятор був ефективнішим, для приміщення №1 (меншого розміру) економія теплоносія становила 4.9%, для приміщення №2 (більшого розміру) 6.8%. Графік споживання енергоносія для обох систем показано на рис. 3.25.

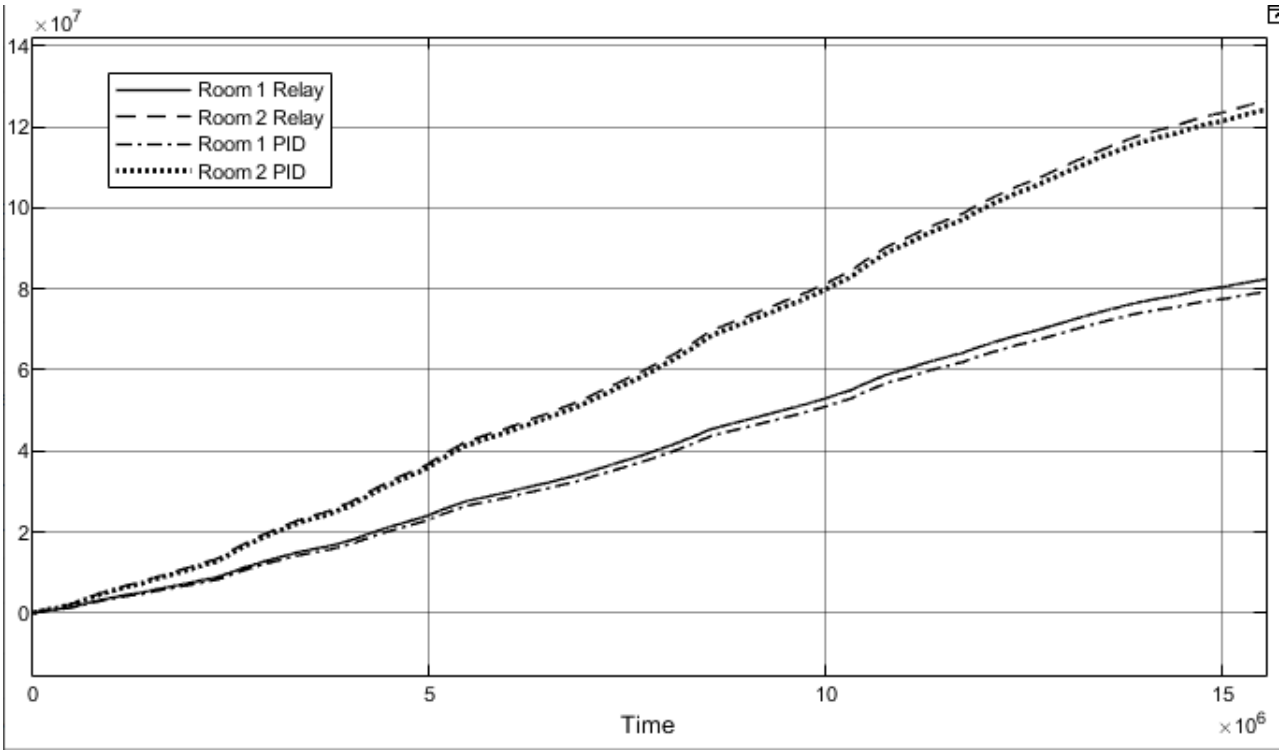


Рис. 3.25 Графік споживання енергоносія протягом опалювального сезону

На рис. 3.26 та рис. 3.27 показано графік зміни температур в приміщеннях та температур радіаторів в приміщенні протягом опалювального періоду для системи з PID регулятором та системи з термостатом відповідно.

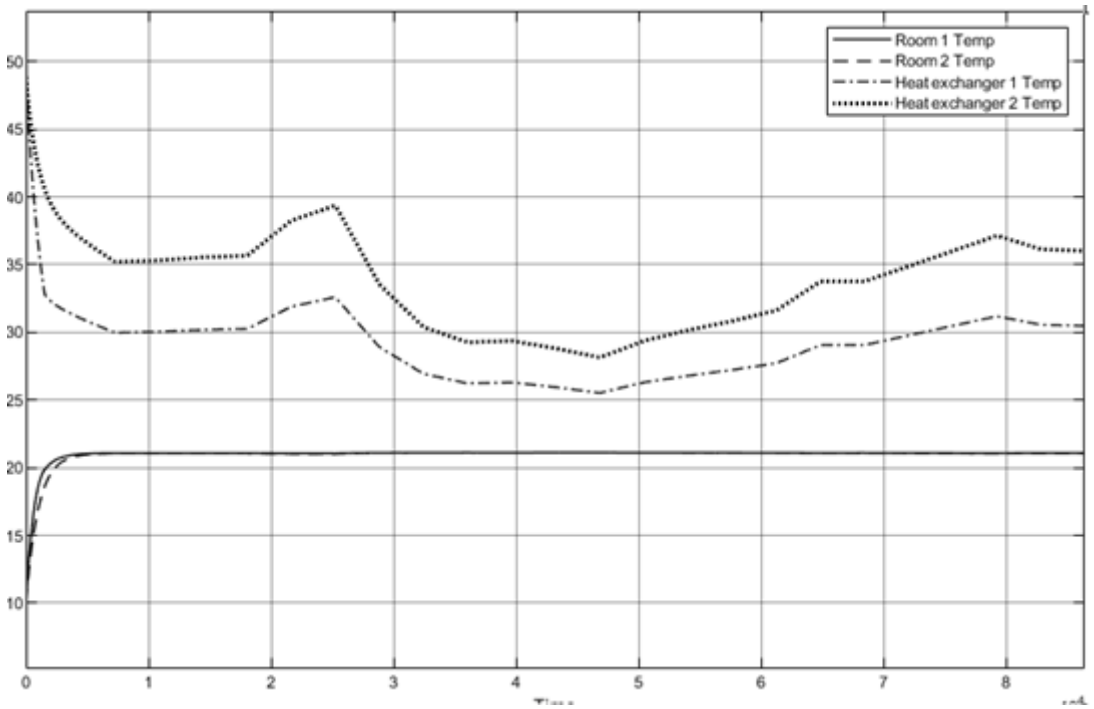


Рис. 3.26 Графік зміни температур в системі з PID регулятором за добу

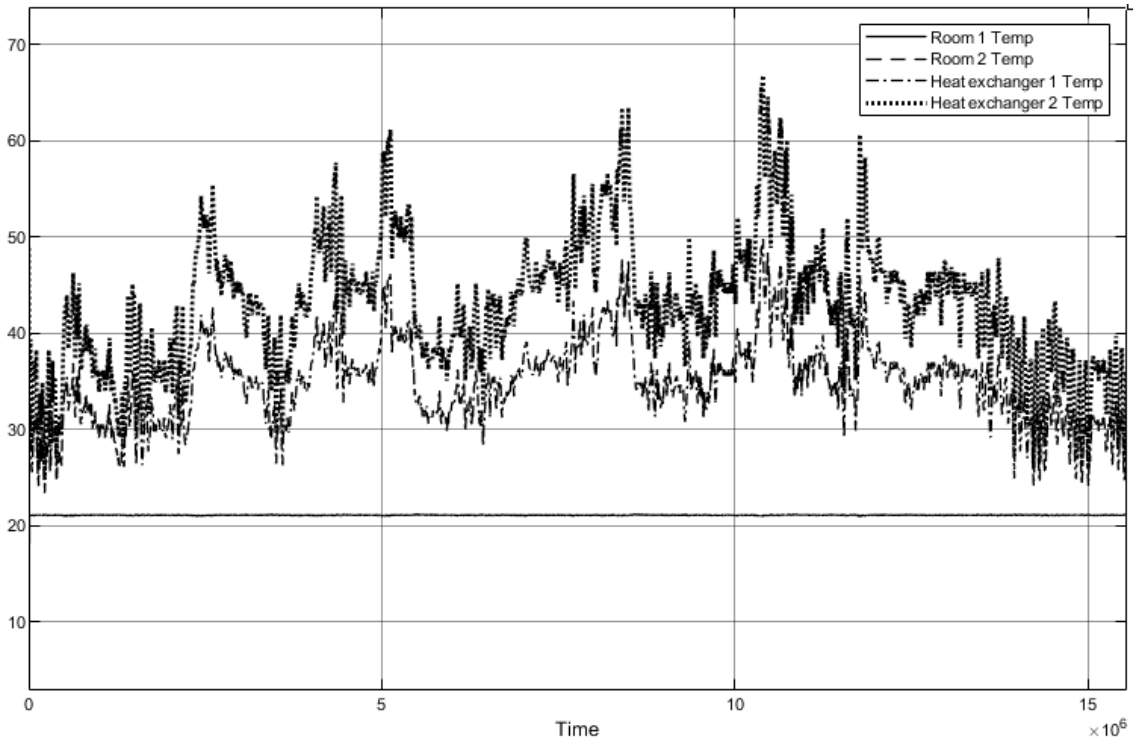


Рис. 3.27 Графік зміни температур в системі з PID регулятором за весь опалювальний сезон

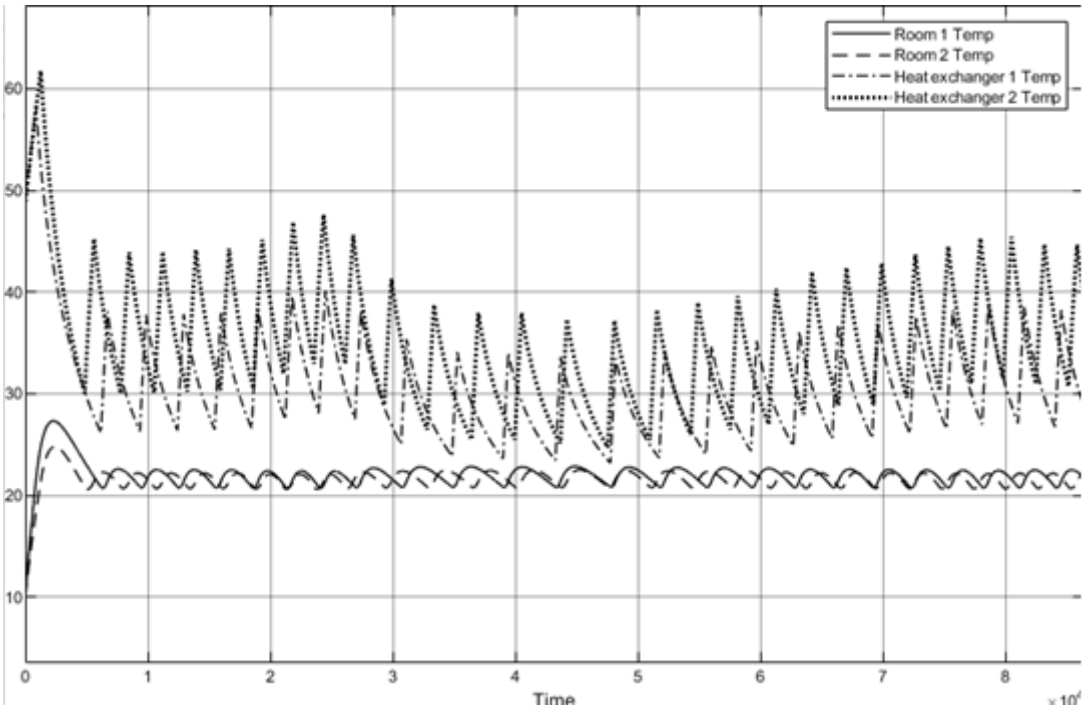


Рис. 3.28 Графік зміни температур в системі з термостатом за добу

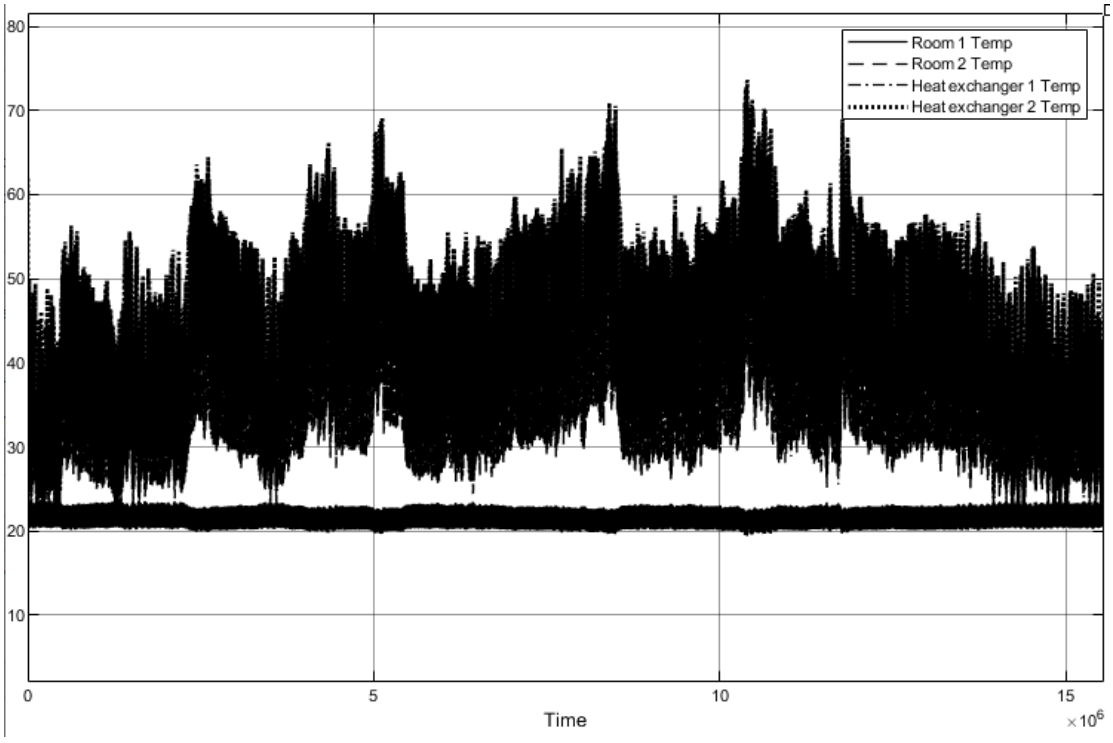


Рис. 3.29 Графік зміни температур в системі з термостатом за весь опалювальний сезон

3.2 Розроблення SCADA системи

SCADA система є основою верхнього рівня автоматизації, призначена для розробки або забезпечення роботи в реальному часі систем збору, обробки, відображення та архівування інформації про об'єкт моніторингу або управління.

SCADA-системи використовують як і у складних технологічних процесах, де потрібно забезпечувати операторський контроль за процесами в реальному часі, так і у відносно простих системах автоматизації, таких як керування системами водопостачання, опалення, охорони будинку. Це програмне забезпечення встановлюється на комп'ютери і, для зв'язку з об'єктом, використовує драйвери введення-виведення або OPC/DDE сервери. Програмний код може бути як написаний на одній з мов програмування, так і згенерований в середовищі проектування. Іноді SCADA-системи комплектуються додатковим ПО для програмування промислових контролерів.

Найбільш широко поширене розуміння SCADA як додатки, тобто програмного комплексу, що забезпечує виконання зазначених функцій, а також інструментальних засобів для розробки цього програмного забезпечення.

В даній роботі в якості програмного забезпечення SCADA використовується Wonderware InTouch. Для моделювання роботи об’єкту – двозонного приміщення використовується Simulink. Зв’язок між SCADA системою та Simulink буде реалізовано двома способами:

- з використанням окремого OPC серверу:
- з використанням протоколу SuiteLink розробленого компанією Wonderware та OPC серверу що наявний в InTouch за замовчуванням.

OPC - це набір специфікацій стандартів для промислових засобів зв'язку. Стандарт OPC описує набір функцій обміну даними в реальному часі між контрольними приладами різних виробників. Стандарт OPC був створений з

метою забезпечення сполучного "моста" між додатками, написаними під Windows, і апаратними засобами, що забезпечують автоматичне керування обробкою даних. Основна мета - скорочення повторюваних дій виробників апаратного забезпечення.

Багато з OPC протоколів базуються на Windows-технологіях: OLE, ActiveX, COM / DCOM. Такі OPC протоколи, як OPC XML DA і OPC UA, є платформи незалежними. OPC DA (Data Access) — основний і найбільш затребуваний стандарт. Описує набір функцій обміну даними в реальному часі з ПЛК та іншими пристроями.

Програмне забезпечення Matrikon OPC DA Server буде використано в якості сервера та Matrikon OPC Explorer для перевірки роботи сервера та системи вцілому.

Для звязку MATLAB з OPC сервером буде використано OPC Toolbox. OPC Toolbox забезпечує доступ до поточних та історичних даних OPC безпосередньо з MATLAB та Simulink, дає можливість читати, записувати та реєструвати дані OPC з таких пристроїв, як розподілені системи управління, системи контролю та збору даних та програмовані логічні контролери. OPC Toolbox дозволяє працювати з даними з серверів які відповідають стандарту OPC Data Access (DA), стандарту OPC Historical Data Access (HDA) та стандарту OPC Unified Architecture (UA). В даній роботі буде використано стандарт OPC DA.

SuiteLink - це протокол зв'язку компанії Wonderware, заснований на TCP / IP, і розроблений спеціально для задоволення виробничих потреб, таких як цілісність даних, висока пропускна здатність та простіша діагностика. Сервер SuiteLink забезпечує передачу даних у середовищі Windows NT для зв'язку між компонентами Wonderware FactorySuite. SuiteLink підтримує властивості даних (VTQ) для Value, Time Stamp та Quality, які важливі для обробки алармів, історичного архівування та SCADA систем. SuiteLink - протокол зв’язку,

розроблений Wonderware, щоб дозволяє програмам, що працюють на ОС Windows, надсилати та отримувати дані. Концепція SuiteLink заснована на архітектурі клієнт / сервер. Серверна програма - сервер Woodhead PCDDE - надає дані і відповідає на запити даних від усіх клієнтів, таких як Wonderware FactorySuite, Archestra IAS (Industrial Application Server). Сервер PCDDE може бути розташований або на локальній машині, або на віддалених машинах.

SCADA-система, розроблена в даній роботі виконує наступні функції:

- відображення значень регульованих параметрів;
- реєстрація алармів;
- відображення поточних трендів;

Основне вікно системи дає змогу задавати необхідну температуру в приміщенні, бачити поточну температуру в приміщенні, температуру зовнішнього повітря.

3.2.1 Зв’язок між SCADA системою та Simulink за допомогою OPC сервера

Схему обміну даними між середовищем Simulink та SCADA системою InTouch через OPC сервер зображено на рис. 3.28.

Алгоритм налаштування з’єднання між MATLAB та SCADA системою показано на простому прикладі з використанням двох змінних. Спочатку було створено Simulink модель з блоками OPC Config, OPC read та OPC Write яку зоражено на рис. 3.29. Налаштування блоків OPC Config, OPC read та OPC Write наведено на рис. 3.30, 3.31, 3.32 відповідно.

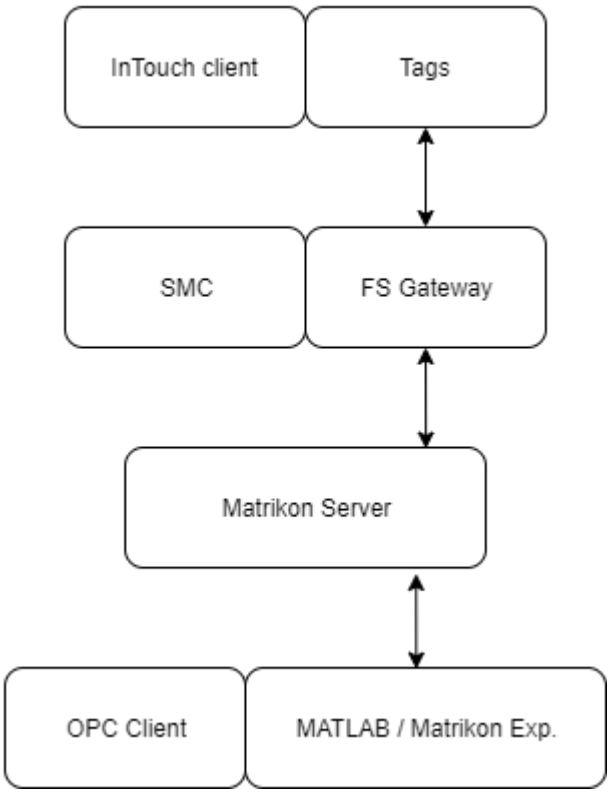


Рис. 3.28 Схема обміну даними

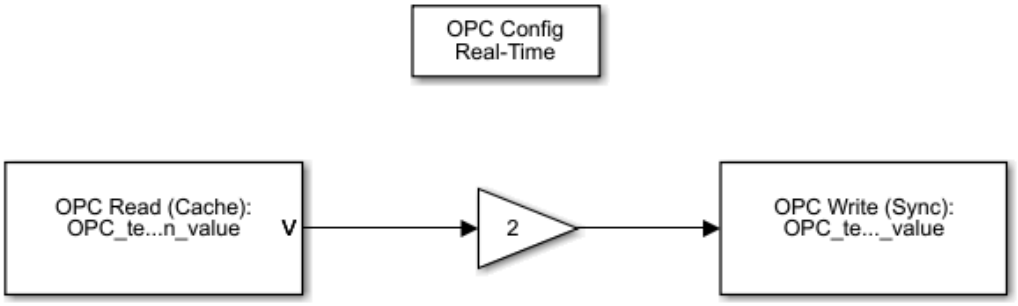


Рис. 3.29 Simulink модель для демонстрації роботи Simulink зі SCADA системою

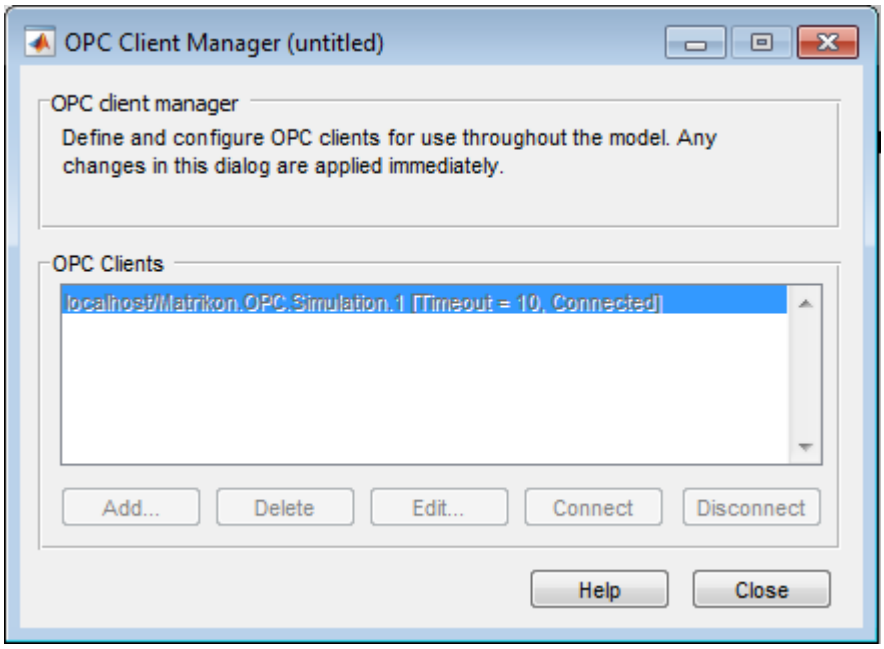


Рис. 3.30 Налаштування блоку OPC Config з вибором OPC серверу

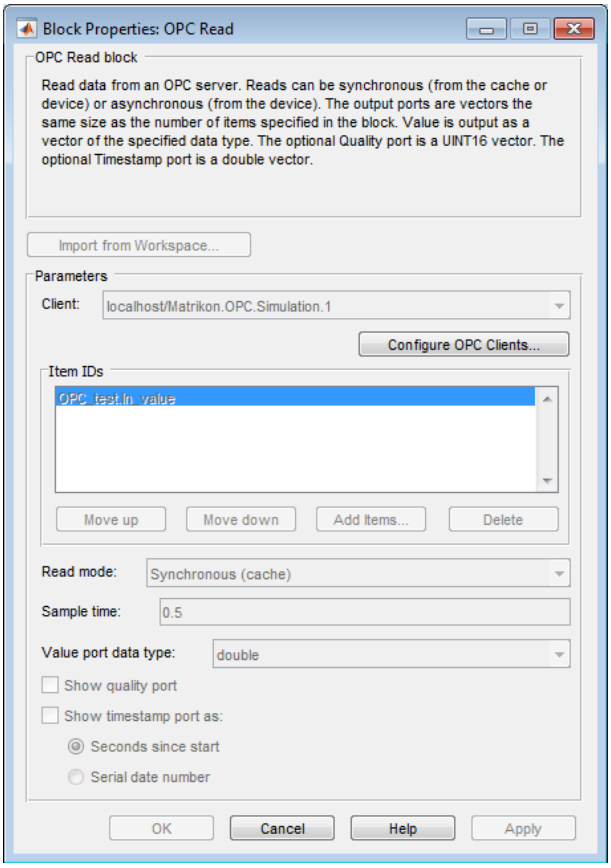


Рис. 3.31 Налаштування блоку OPC Read

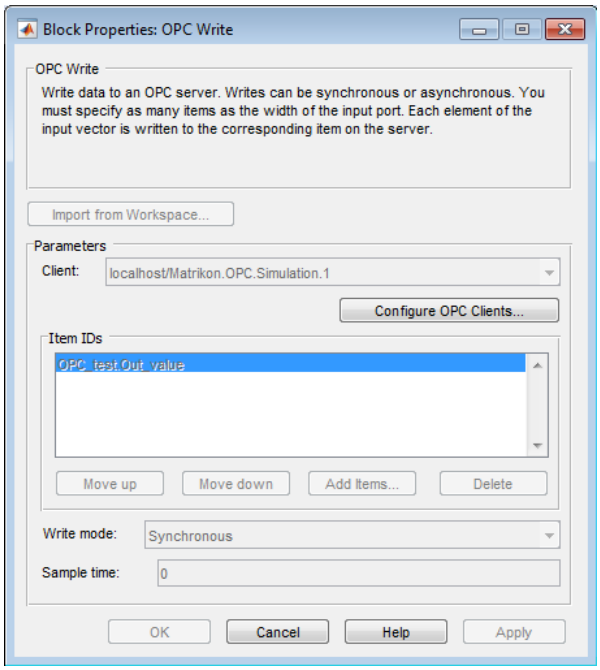


Рис. 3.32 Налаштування блоку OPC Write

Далі запущено OPC Server конфігурація якого заображена на рис. 3.33.

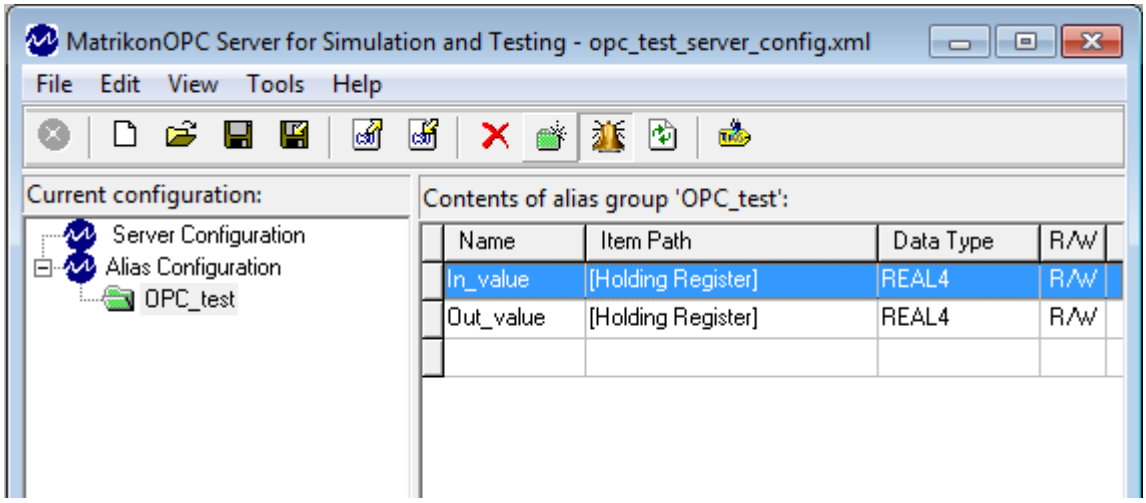


Рис. 3.33 Конфігурація OPC сервера

Перевірку роботи сервера було виконано за допомогою програмного забезпечення Matrikon OPC Explorer. Наявні на сервері теги зображено на рис. 3.34.

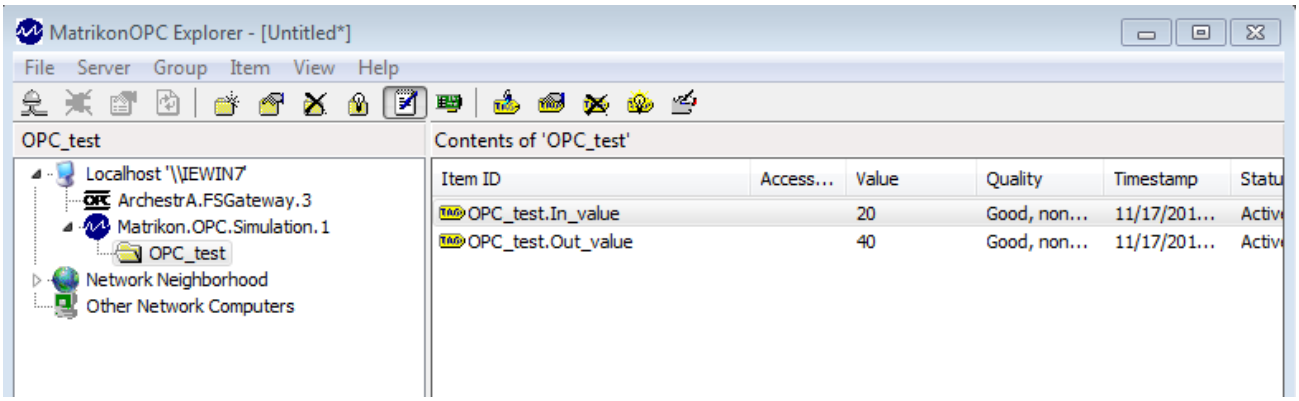


Рис. 3.34 Наявні на сервері теги

Для того щоб теги наявні на OPC сервері були доступні в SCADA системі потрібно налаштувати шлюз ArchestraFSGateway, а саме створити об’єкт OPC в SMC із заданням потрібного OPC сервера (рис. 3.35) та додати необхідні теги в Device Items (рис. 3.36).

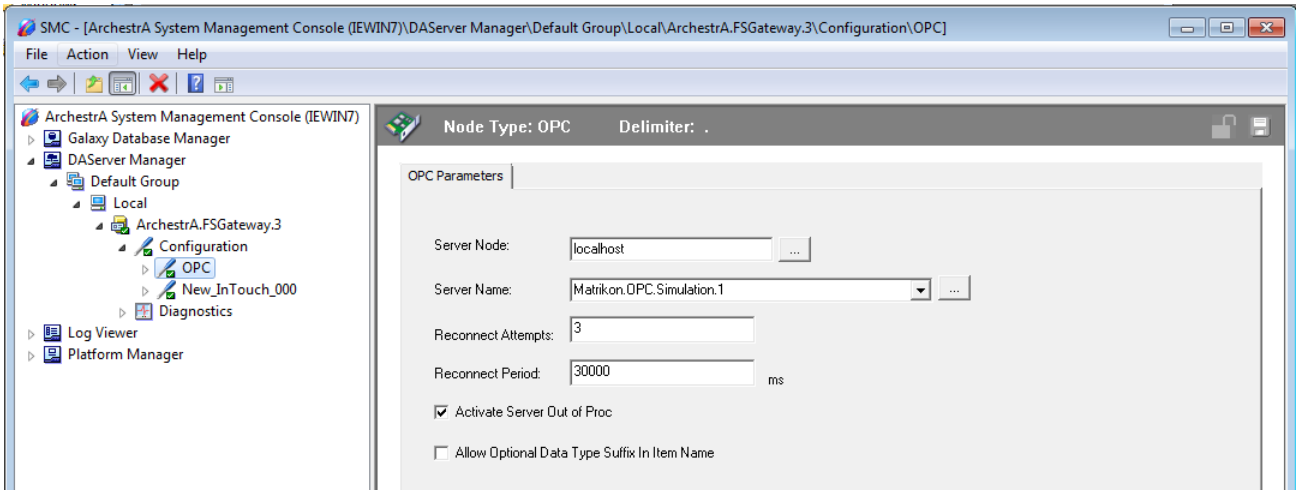


Рис. 3.25 Об’єкт OPC в шлюзі ArchestraFSGateway

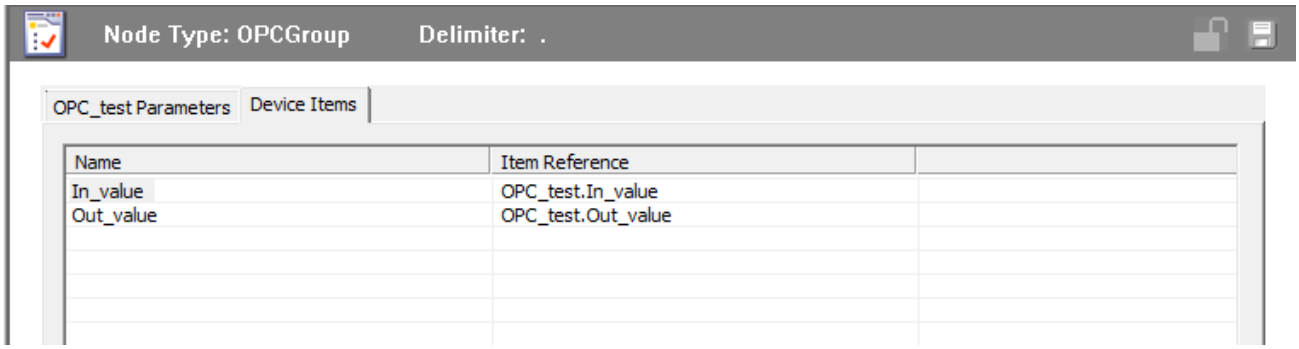


Рис. 3.36 Додані теги в Device Items OPC об’єкта

Для того щоб бачити як змінюються значення тегів в InTouch, створено вікно Main з двома контролами, один із яких дає можливість задати один із тегів доступний на сервері, а інший відображає значення другого тегу отримане з серверу (рис. 3.37).

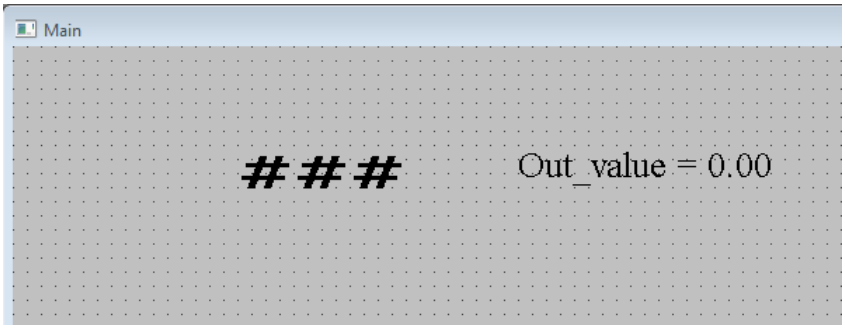


Рис. 3.37 Вікно Main що показує значення тегів

Щоб теги були доступні в InTouch потрібно створити Access Name (рис. 3.38) та додати необхідний тег в Tagname Dictionary (рис. 3.39).

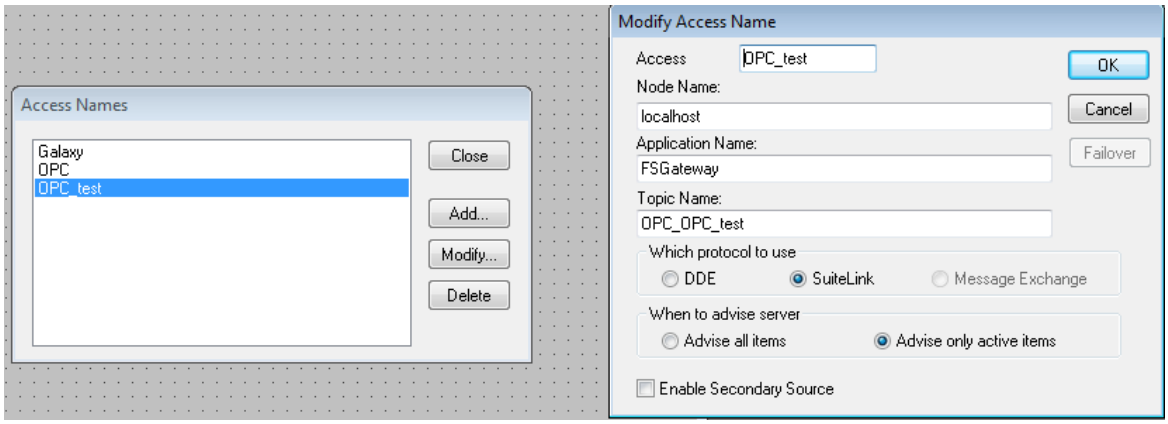


Рис. 3.38 Створення Access Name в InTouch

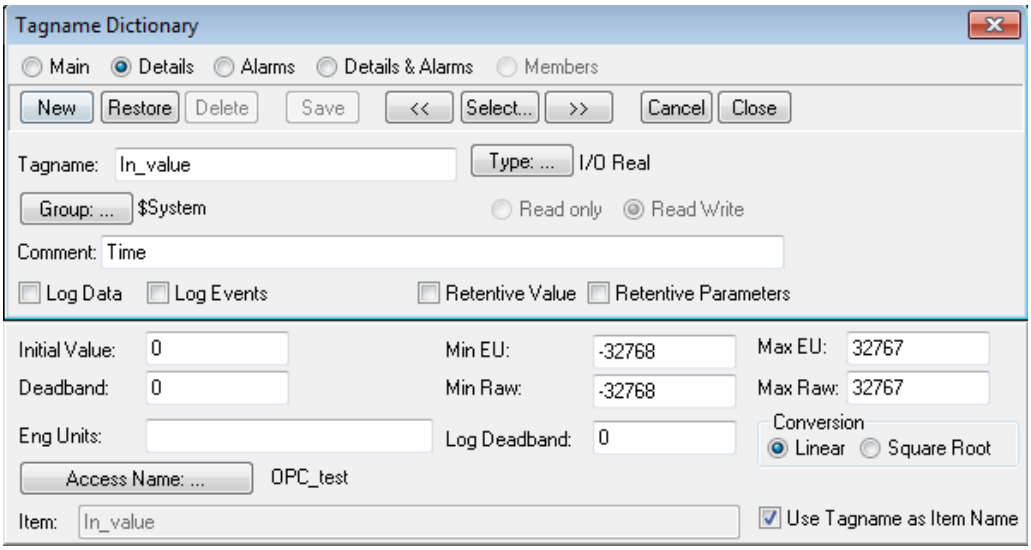


Рис. 3.39 Створення тегу в Tagname Dictionary

Після виконання всіх наведених вище інструкцій, після переходу в режим Runtime системи отримали результат зображений на рис. 3.40.

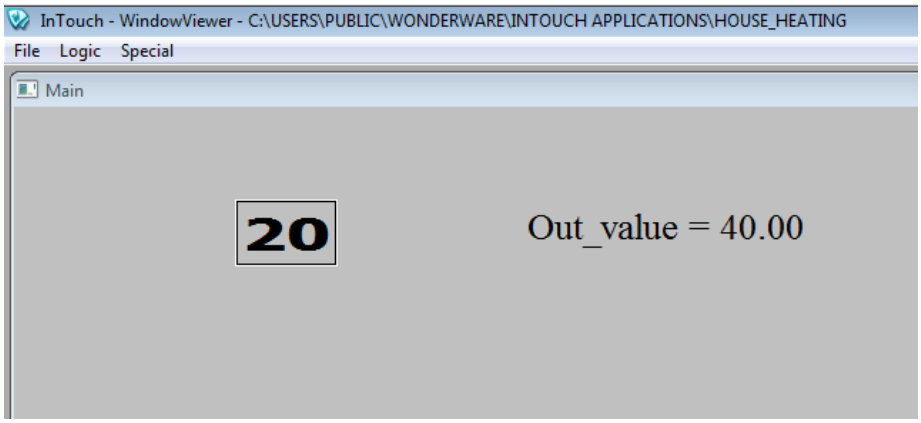


Рис. 3.40 Вікно що показує результат роботи SCADA системи та Simulink

3.2.2 Зв'язок між SCADA системою та Simulink через протокол SuiteLink компанії Wonderware та OPC серверу

Схему обміну даними між середовищем Simulink та SCADA системою InTouch через протокол SuiteLink зображено на рис. 3.41.

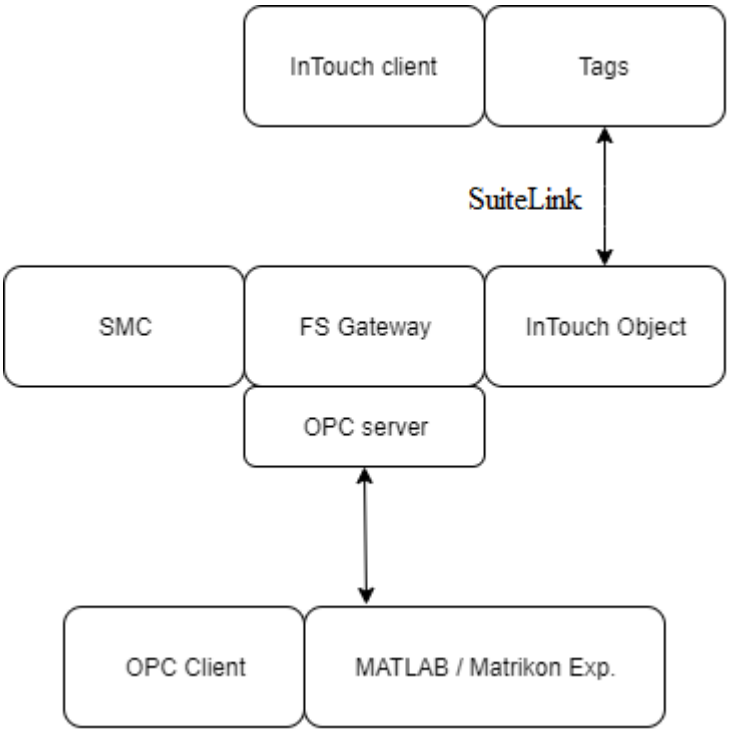


Рис. 3.41 Схема обміну даними

Алгоритм налаштування з'єднання між MATLAB та SCADA системою в даному випадку не суттєво відрізняється від алгоритму наведеного в пункті 3.2.1.

Модель Simulink та налаштування блоків виконуються аналогічно. Для того щоб скористатись функціоналом вбудованого в InTouch OPC сервера та протокола Suitelink необхідно додати в SMC в шлюз ArchestraFSGateway об'єкт InTouch (рис. 3.42) та додати теги в Device Items об'єкта (рис. 3.43).

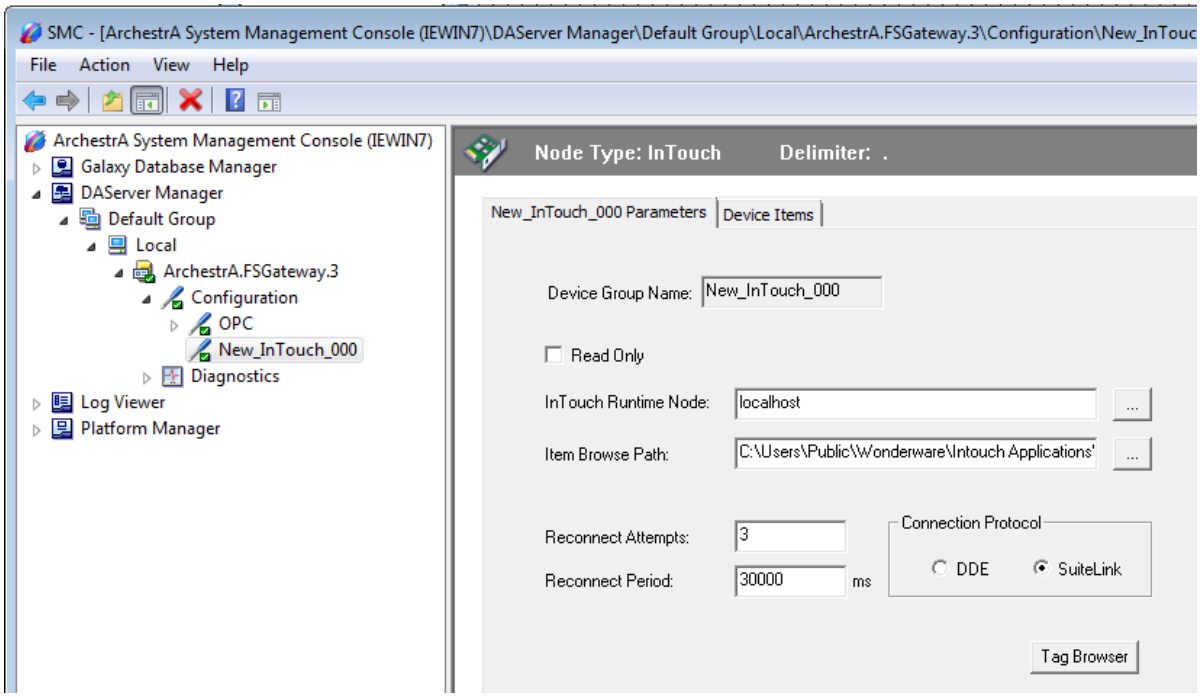


Рис. 3.42 Створення об’єкту InTouch в SMC

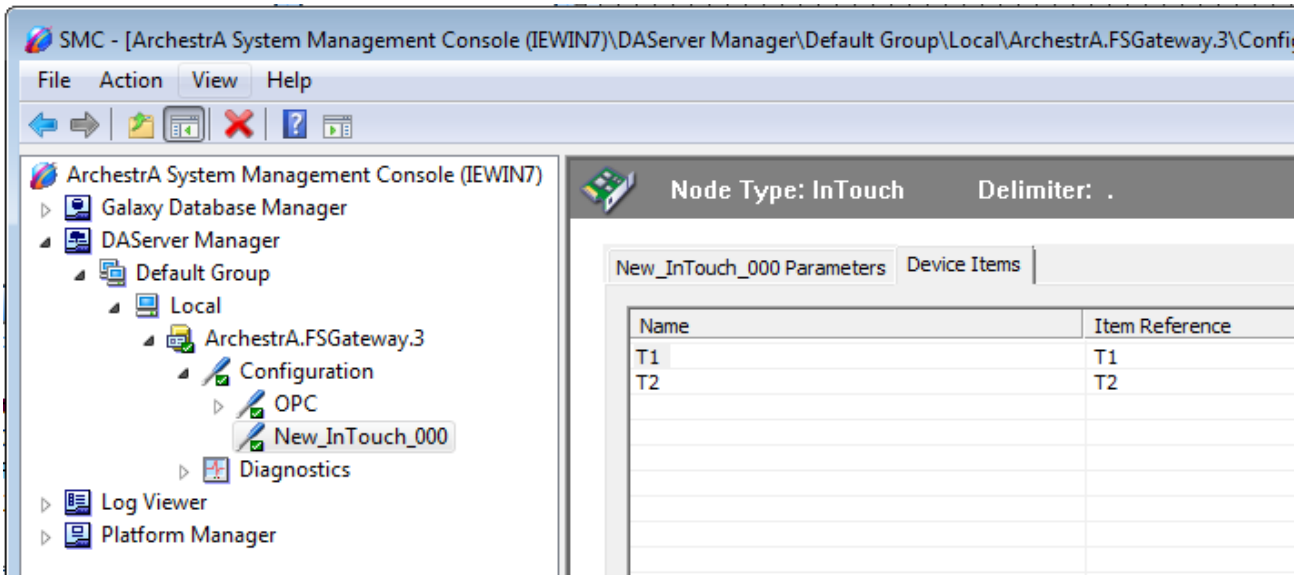


Рис. 3.43 Додані теги в Device Items об’єкта InTouch

На рис. 3.44 виконано перевірку виконаних дій за допомогою Matrikon OPC Explorer, теги додані в об’єкт InTouch активні на сервері ArchestrA.

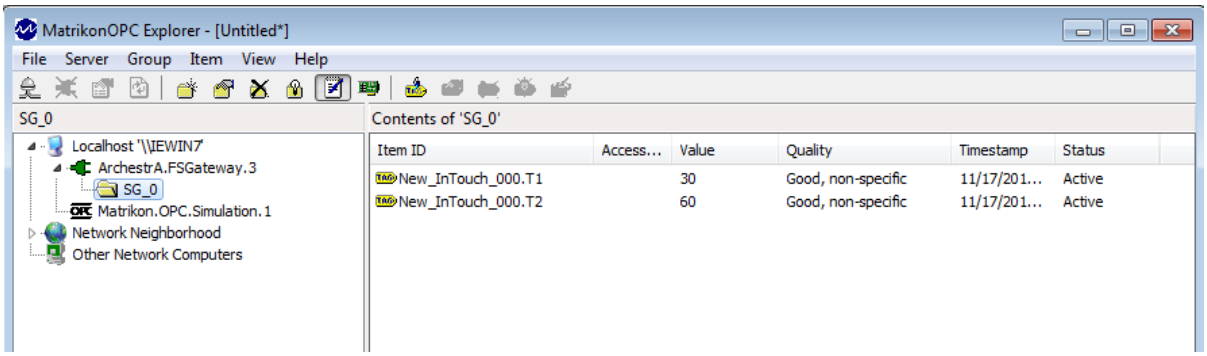


Рис. 3.44 Наявні на сервері теги

Далі для доступу до даних тегів в InTouch потрібно лише створити Access Name по аналогії як це було зроблено в пункті 3.2.1 та створити теги в Tagname Dictionary (рис. 3.45). На рис. 3.46 показано запущену систему в якій зв'язок з MATLAB працює з використанням вбудованого в InTouch OPC сервера.

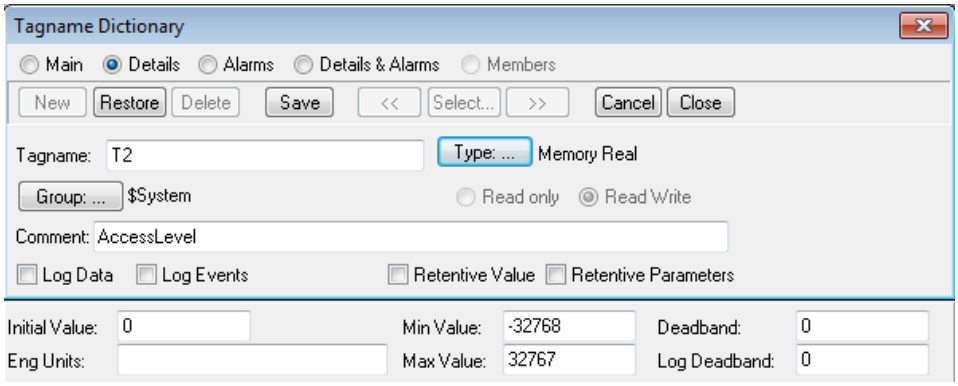


Рис. 3.45 Створення тегу в Tagname Dictionary

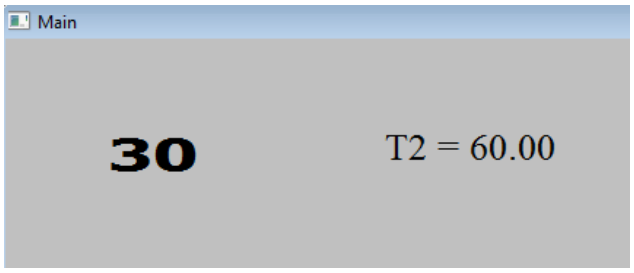


Рис. 3.46 Вікно що показує результат роботи SCADA системи та Simulink

3.2.3 Розроблення та зв'язок SCADA системи обігріву приміщення з Simulink моделлю

Під час створення системи буде використано спосіб передачі даних з Simulink моделі в SCADA систему описаний в пункті 3.2.2 так як він не потребує додаткового використання окремого OPC серверу.

Першим кроком в розробці буде створення інтерфейсу SCADA системи після чого буде зрозумло які теги потрібні для роботи системи та які дані з Simulink моделі читати/писати на OPC сервер. На рис. 3.47 зображено розроблене головне вікно системи, яке відображає поточні значення температур в приміщенні, поточне значення температури на вулиці, дозволяє включити/виключити систему керування, та дозволяє задати необхідні значення температур в приміщенні

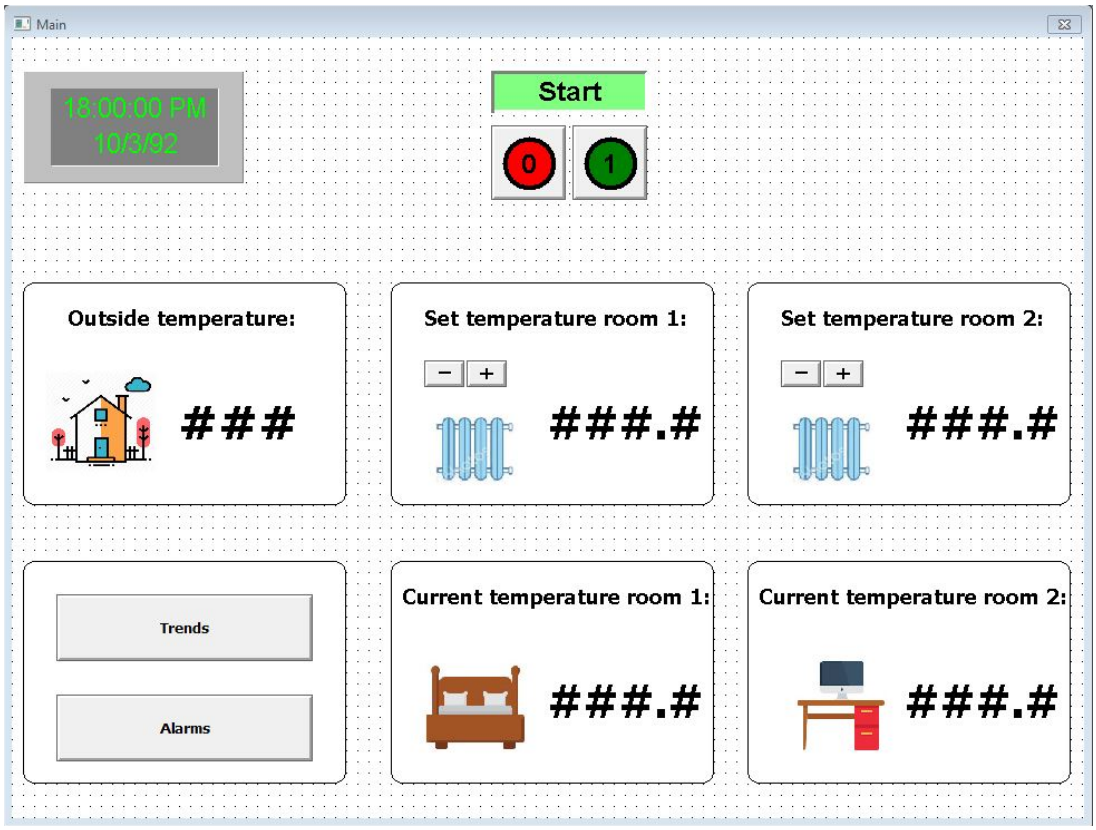


Рис. 3.47 Головне вікно SCADA системи

Після того як були визначені всі необхідні функції системи та набір даних яким потрібно обмінюватись з Simulink моделлю, було створено InTouch об’єкт в SMC та додано необхідні теги (рис. 3.48).

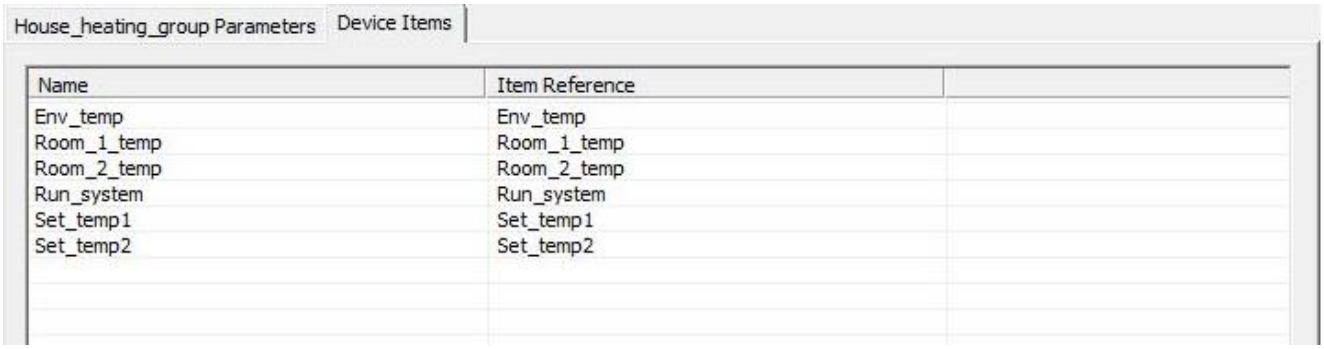


Рис. 3.48 Теги в Device Items необхідні для роботи системи

Перевірку доступності тегів на OPC сервері ArchestrA було виконано за допомгою Matrikon OPC Explorer (рис. 3.49)

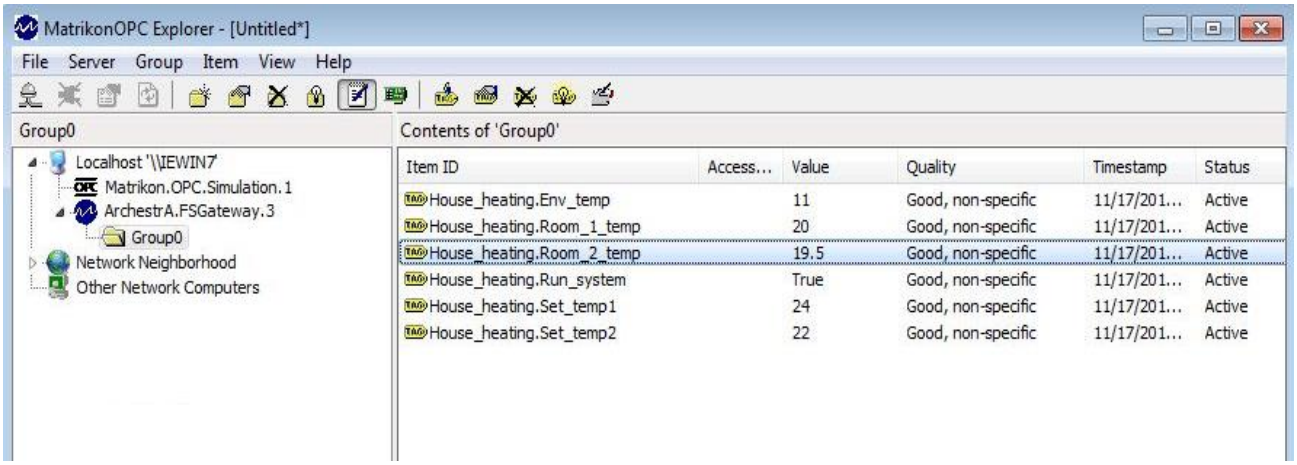


Рис. 3.49 Наявні на сервері ArchestrA теги

Для зчитування та запису даних з Simulink моделі використано OPC Toolbox, роботу з елементами якого було описано в пункті 3.2.1. Для роботи системи необхідно писати на OPC сервер поточне значення температури повітря на вулиці, поточні значення температур в приміщенні та зчитувати з серверу значення змінної що відповідає за включення системи керування, значення заданих температур в приміщенні.

На рис. 3.50 зображено Simulink модель системи з блоками запису/зчитування даних з OPC серверу. На рис. 3.51 головне вікно системи під час роботи.

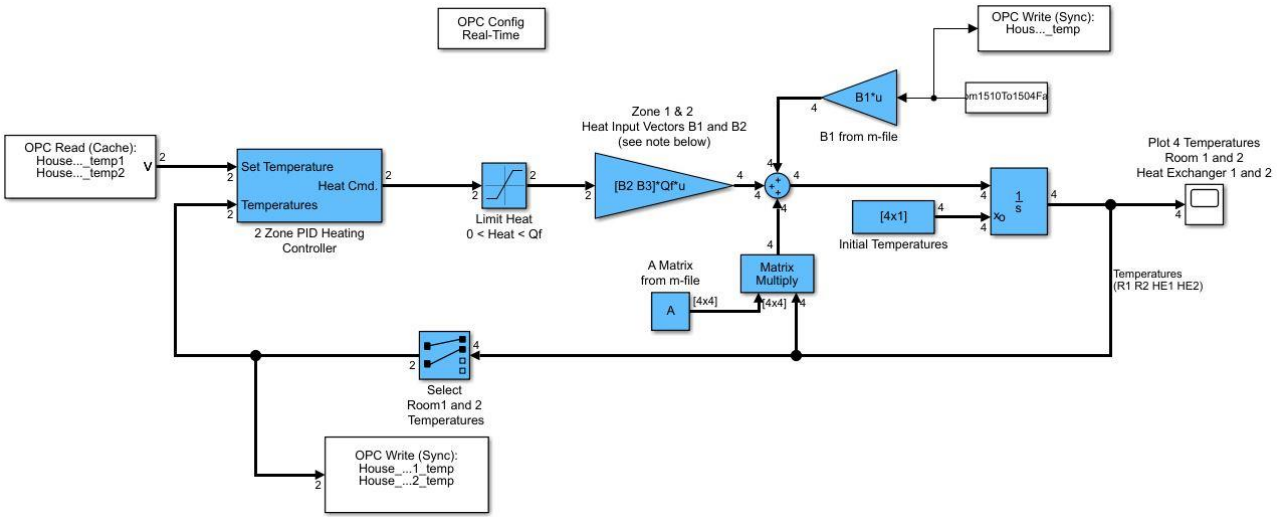


Рис. 3.50 Simulink модель з передачею даних на OPC сервер

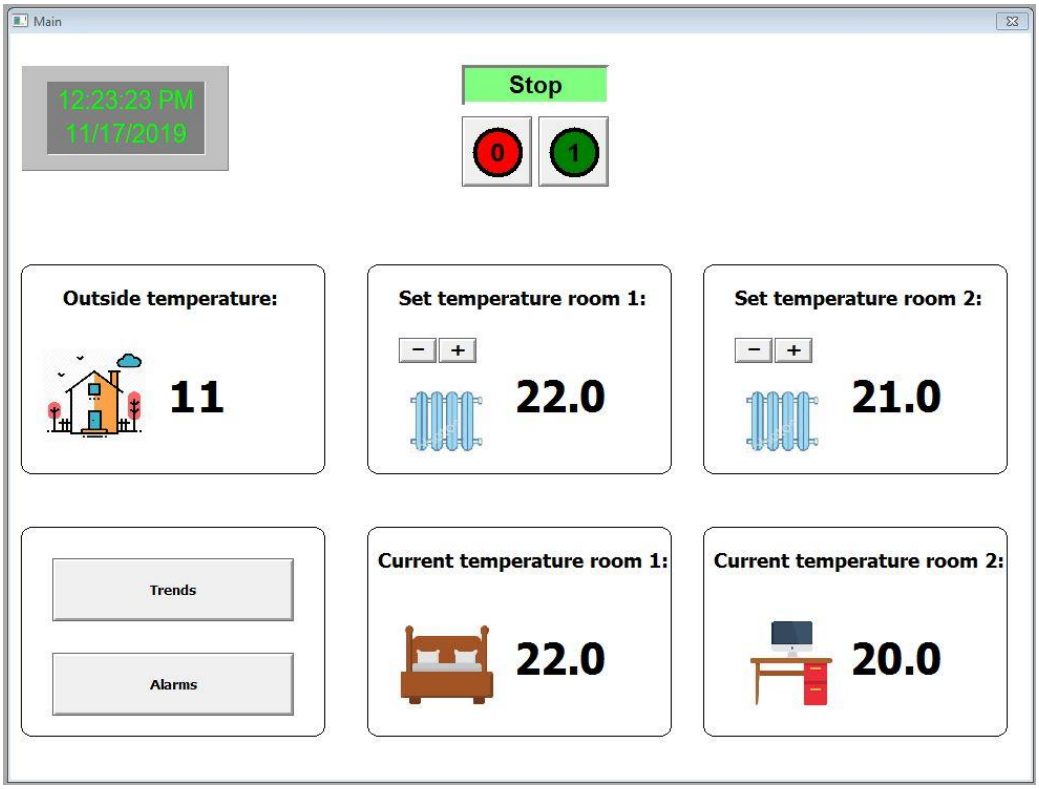


Рис. 3.51 Головне вікно системи під час роботи

На рис. 3.52 та 3.53 зображено вікно трендів та вікно алармів системи відповідно.

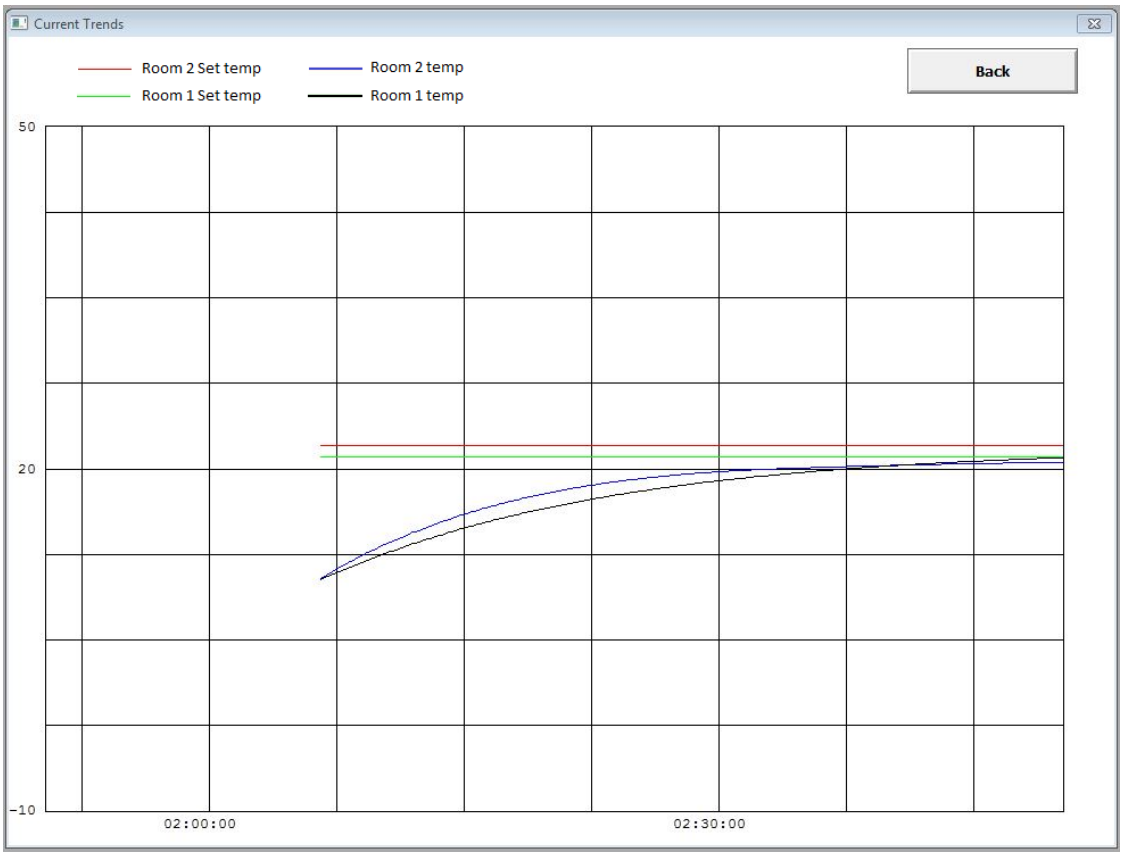


Рис. 3.52 Вікно трендів системи

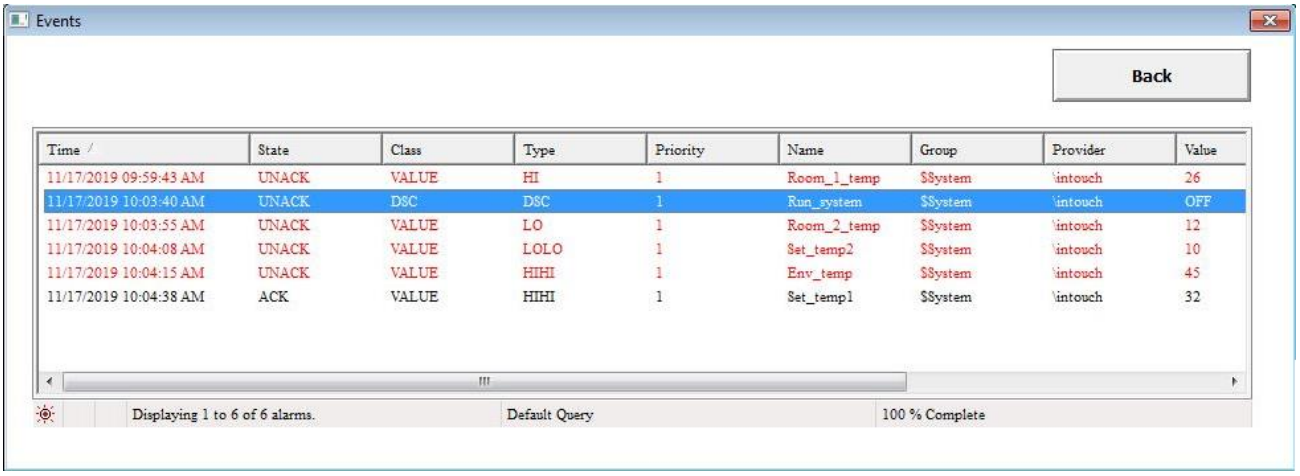


Рис. 3.53 Вікно алармів системи

РОЗДІЛ 4
РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Завданням компанії є виробництво контролерів для опалення будинку. Завдяки використанню сучасних технологій компанія, якості виготовленого товару та використаного програмного забезпечення буде заробляти рейтинг обслуговування клієнтів, який буде перевищувати конкурентів. Знання населення про відмінність в роботі PID регулятора та термостата є недостатнім. Мало хто знає, що PID регулятор є ефективнішим. Наша місія полягає в тому, щоб подолати розрив у рівні знань.

Таблиця 1. Опис ідеї стартап-проекту

Зміст ідеї	Напрями застосування	Вигода для користувача
Розробка контролерів для опалення багатозонних приміщень з використанням середовища Simulink та бібліотеки Stateflow для	1.Програмування контролерів в середовищі Simulink з використанням бібліотеки Stateflow	Зменшення витрат часу спеціаліста з автоматизації на розробку, відлагодження та тестування програмного забезпечення
	2.Встановлення контролерів в приватних будинках та комерційних приміщеннях	Збільшення ефективності керування опаленням в будинку як з точки зору створення теплового комфорту так і з точки зору використання енергоносія

4.2 Технологічний аудит ідеї проекту

Загальна наявність та доступність потрібних для реалізації проекту технологій наведена в таблиці 2.

Таблиця 2 Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Розробка контролерів для опалення багатозонних приміщень та їх встановлення в приміщеннях	Створення програмного коду буде виконуватись в середовищі Simulink з використанням бібліотеки Stateflow. Код буде загружатись на контролер що підримує мову С	Середовище для розробки наявне. Програмний код та модель для тестування для системи керування двозонним приміщенням наявні. Потрібно створити універсальний алгоритм керування для приміщення з багатьма зонами. Контролер що підтримує мову С – наявний.	Для використання середовища MATLAB та Simulink потрібні ліцензії. Бібліотека Stateflow та все інше програмне забезпечення знаходиться у вільному доступі. Контролери що підтримує мову С знаходяться у вільному продажі.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Наявність попиту, обсяг, динаміка розвитку ринку наведено в таблиці 3.

Таблиця 3 Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	>20
2	Загальний обсяг продаж, грн/ум.од	>1 000 000$
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Велика кількість конкурентів, необхідно забезпечити стабільний ринок збуту
5	Специфічні вимоги до стандартизації та сертифікації	Аналогічні по галузі
6	Середня норма рентабельності в галузі (або по ринку), %	15%

Потенційні групи клієнтів, їх характеристики та орієнтовний перелік вимог до товару для кожної групи наведено в таблиці 4.

Таблиця 4. Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Збільшення ефективності керування опаленням в будинку як з точки зору створення теплового комфорту так і з точки зору використання енергоносія	Власники приватних будинків, власники квартир елітного сегменту що мають досить велику площу, власники комерційної нерухомості які мають дохід вище середнього. Компанії забудовники	Всі групи клієнтів однаково зацікавлені в якості та ефективності керування опаленням в приміщенні. Компанії забудовники потребують універсального рішення, що може бути використане як і в багатоквартирному будинку так і в приватному.	Вимоги до продукції: надійність та ефективність. Вимоги до компанії: наявність ефективної системи підтримки та допомоги під час експлуатації системи.

4.4 Розроблення ринкової стратегії проекту

Опис цільових груп потенційних споживачів наведено в таблиці 5.

Таблиця 5 Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Власники будинків, комерційних приміщень	Висока	Середня	Висока	Середня
Компанії забудовники	Висока	Середня	Висока	Середня

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 6).

Таблиця 6 Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Основні конкурентноспроможні позиції	Базова стратегія розвитку
Розповсюдження послуги та продукту	Стратегія диференційованого маркетингу	Ефективність керування системою опалення	Стратегія спеціалізації

Наступним кроком є вибір стратегії конкурентної поведінки.

Таблиця 7. Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Проект не є першопрохідцем	Компанія буде шукати нових та забирати частину незадоволених клієнтів у конкурентів	Компанія буде копіювати інтерфейс керування системою опалення приміщення	Стратегія наслідування лідеру

В результаті маємо узгоджену систему рішень, що визначатиме напрями роботи стартап-компанії на ринку.

4.5 Розроблення маркетингової стратегії

Метою маркетингового плану є розробка посібника щодо того, як компанія буде будувати відносини з клієнтами протягом своєї діяльності. Це буде виконано шляхом збереження високого рейтингу обслуговування клієнтів, одночасно досягаючи цілей, які співвідносяться з нашою місією. Через рекламу на дошках оголошень, місцеву рекламу, рекомендації від забудовників, особисту розсилку повідомлень дасть високий рівень обізнаності по всій території України. З плином часу компанія вибудує бренд і збере список клієнтів. Цей список буде містити клієнтів, які дали відгуки про нашу роботу після завершення

встановлення. Потім компанія буде використовувати результати опитування як спосіб забезпечити якісну роботу іншим потенційним клієнтам.

Таблиця 8. Рекламні витрати компанії

Рекламні витрати	
Вид реклами	$
Інтернет дошки оголошень	100
Місцева реклама	500
Збір бази даних потенційних клієнтів та розсилка особистих повідомлень	4 000
Реклама від забудовників	2 000
Власний веб-сайт та вся необхідна інформація	5 000

Поточні клієнти є нашим основним інструментом у виявленні ефективності маркетингового плану. Клієнти можуть розповісти про те, як вони почули про компанію і чи порадили б вони нашу компанію комусь іншому. Ми можемо отримати відгуки від наших клієнтів і запитати, наскільки вони задоволені обслуговуванням та їх новою системою керування опаленням. Ми можемо відстежувати наші конкретні маркетингові зусилля, визначаючи, наскільки успішними ми є у здобутті клієнтів, підтримці клієнтів та побудові надійного бренду. Ці зусилля будуть очевидні після того, як маркетинговий план буде у використані протягом певного часу. Ці знання допоможуть нам зрозуміти, де зосередити більше ресурсів і де їх зменшити.

На рис. 4.1 показані операційні бізнес-процеси компанії.

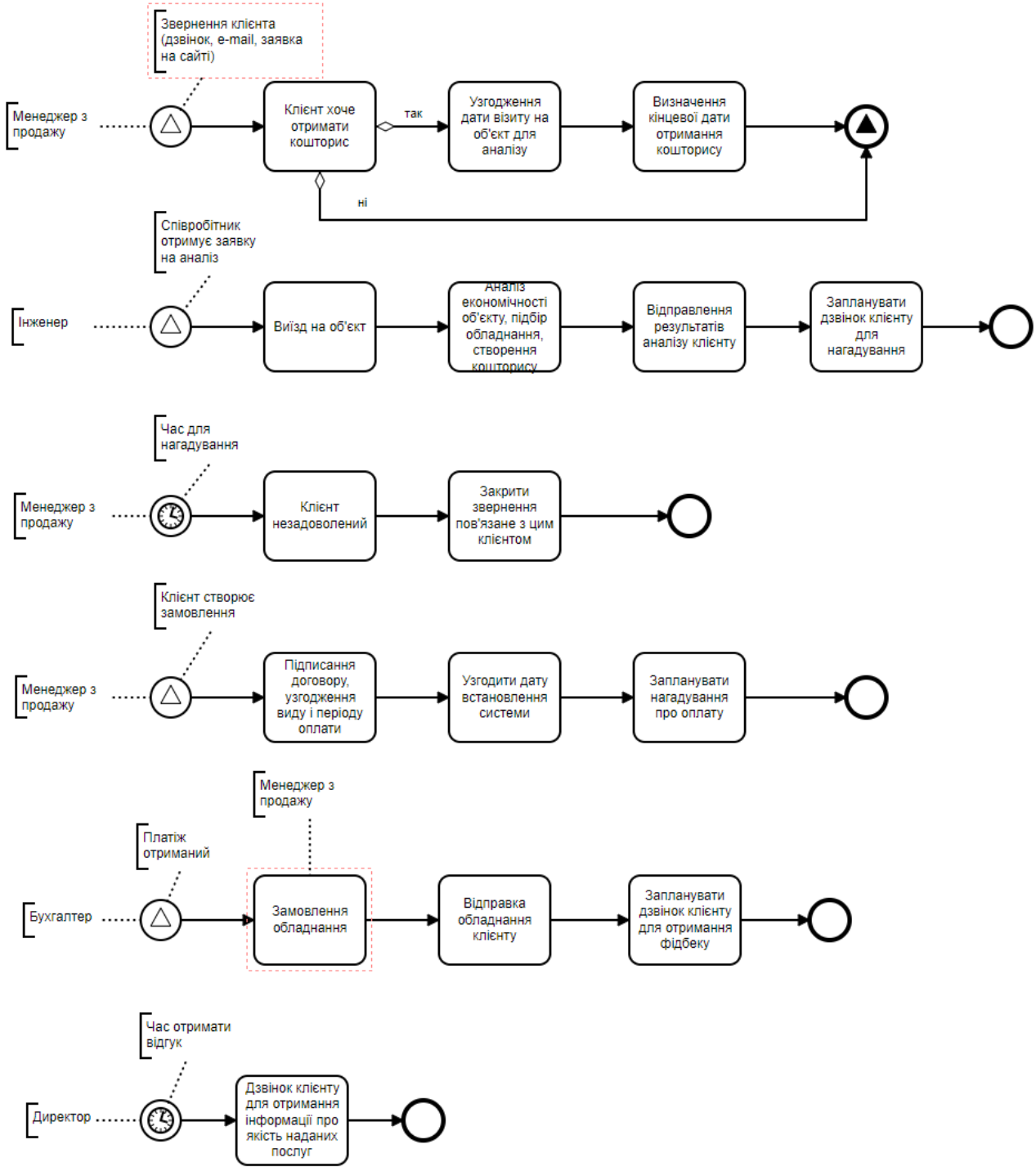


Рис. 4.1 Бізнес-процеси компанії

ВИСНОВКИ

Отже, в результаті виконання даної магістерської дисертації отримано наступне:

1. Створено та реалізовано модель системи автоматизованого керування обігрівом двозонного приміщення для всього спектру режимів роботи (протягом всього опалювального сезону) в середовищі Simulink із застосування бібліотеки Stateflow. Для моделювання були підготовлені та використані реальні метеодані для міста Київ, значення температури змінюється погодинно протягом всього опалювального сезону (з 15 жовтня по 15 квітня). Наведено приклад використання бібліотеки Stateflow для створення діаграми керування фізичним контролером обігріву в приміщенні з якої в подальшому згенеровано код для використання в контролерах що підтримують мови програмування C/C++.
2. Розроблено SCADA-систему на основі програмного забезпечення Wonderware InTouch у взаємозвязку із середовищем Simulink через OPC сервер. Наведено два способи передачі даних між SCADA системою та середовищем Simulink: з використанням окремого OPC сервера та з використанням протоколу SuiteLink. Дані способи обміну даними можна використати як в реальних проектах так і в навчальному процесі, для практичних задач створення SCADA систем, дослідження об’єктів з використанням SCADA систем та інших цілей. Здійснено відлагодження та тестування роботи системи.
3. Виконано порівняння ефективності систем керування на основі PID регулятора та системи з термостатом з точки зору створення комфорту та споживання енергоносія. Система керування на основі PID

регулятора однозначно є кращою з точки зору створення теплового комфорту так як температура в приміщенні підтримується сталою, тоді як з використанням термостату температура постійно відхиляється від заданої. З точки зору ж ефективності використання енергоносія припущення про вищу ефективність системи з PID регулятором підтвердилось. Система з PID регулятором була ефективніша для обох приміщень, для приміщення меншого розміру на 4.9%, для приміщення ж більшого розміру на 6.8%. Моделювання системи з використанням термостата виконувалось із зоною нечутливості 1^0C.

4. Розроблено стартап-проект запуску компанії, що займається виробництвом та продажем ефективних контролерів для опалення приміщень, алгоритм керування яких розробляється з використанням бібліотеки Stateflow.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Положення про організацію дипломного проектування та державної атестації студентів НТУУ "КПІ" / Уклад. В.Ю.Угольніков. За заг.ред. Ю.І.Якименка – К.:ВПК "Політехніка", 2006. – 84 с.
2. Методичні рекомендації щодо оформлення курсових та дипломних проектів для студ. теплоенергетичного факультету усіх спеціальностей очної та заочної форм навчання / Уклад. Т.Г. Баган, С.Г. Батюк, В.П. Бунь, М.Ю. Ізгорев, С.Ю. Олійник. – К.: ІВЦ «Видавництво “Політехніка”», 2001. – 39 с.
3. ДСТУ Б А.2.4-3-95 (ГОСТ 21.408-93) СПДБ. Правила виконання робочої документації автоматизації технологічних процесів.
4. ДСТУ Б А.2.4-4-99 (ГОСТ 21.101-97) СПДБ. Основні вимоги до проектної та робочої документації.
5. Чистович С.А., Аверьянов В.К., Темпель Ю.Я., Быков С. И. Автоматизированные системы теплоснабжения и отопления. – Ленинград. Стройиздат, 1987.
6. Фаликов В.Ф., Витальев В.П. Автоматизация тепловых пунктов: Справочное пособие. - М.: Энергоатомиздат, 1989.
7. Harel, David, Executable Object Modeling with Statecharts, IEEE Computer Society Magazine, Vol. 30, No 2, pp 41-52, April 2002.
8. Lienhard IV, John H., and Lienhard V, John H., A Heat Transfer Textbook, Phlogiston Press, Cambridge, MA, 2002.
9. The MathWorks Inc., Getting Started: Signal Processing Blockset, For Use with Simulink, Reprintm Match 2006.
10. The MathWorks Inc., Users Guide: Stateflow and Stateflow Coder, For Complex Logic and State Diagram Modeling, July 2012.

11. База метеоданих для міста Київ http://www.equaonline.com/iceuser/ASHRAE_IWEC.html

ДОДАТОК А. Програмний код алгоритму керування на мові С згенерований з моделі Stateflow для використання в контролері

```
/*
 * ProcessCommands.c
 *
 * Code generation for model "ProcessCommands".
 *
 * Model version              : 1.86
 * Simulink Coder version : 9.0 (R2018b) 24-May-2018
 * C source code generated on : Mon Dec  2 22:32:55 2019
 *
 * Target selection: grt.tlc
 * Note: GRT includes extra infrastructure and instrumentation for prototyping
 * Embedded hardware selection: 32-bit Generic
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include "ProcessCommands.h"
#include "ProcessCommands_private.h"

/* Named constants for Chart: '<Root>/ProcessCommands' */
#define ProcessComma_IN_NO_ACTIVE_CHILD ((uint8_T)0U)
#define ProcessComman_event_temperature (1)
#define ProcessCommands_IN_AMPM        ((uint8_T)1U)
#define ProcessCommands_IN_Hour1       ((uint8_T)2U)
#define ProcessCommands_IN_Hour2       ((uint8_T)3U)
#define ProcessCommands_IN_Minutes1    ((uint8_T)4U)
#define ProcessCommands_IN_Minutes2    ((uint8_T)5U)
#define ProcessCommands_IN_Output      ((uint8_T)1U)
#define ProcessCommands_IN_Output_Time ((uint8_T)6U)
#define ProcessCommands_IN_SetTemp     ((uint8_T)7U)
#define ProcessCommands_IN_SetTime     ((uint8_T)8U)
#define ProcessCommands_IN_Start       ((uint8_T)2U)
#define ProcessCommands_IN_Start_h     ((uint8_T)9U)
#define ProcessCommands_IN_Temperature ((uint8_T)10U)
#define ProcessCommands_IN_Tens1       ((uint8_T)11U)
#define ProcessCommands_IN_Tens2       ((uint8_T)12U)
#define ProcessCommands_IN_Wait1       ((uint8_T)13U)
#define ProcessCommands_IN_Wait2       ((uint8_T)14U)
#define ProcessCommands_IN_Wait3       ((uint8_T)15U)
#define ProcessCommands_IN_Wait4       ((uint8_T)16U)
#define ProcessCommands_event_next     (4)
#define ProcessCommands_event_run      (0)
#define ProcessCommands_event_tic      (3)
#define ProcessCommands_event_time     (2)

/* Block states (default storage) */
DW_ProcessCommands_T ProcessCommands_DW;

/* Previous zero-crossings (trigger) states */
PrevZCX_ProcessCommands_T ProcessCommands_PrevZCX;
```

```
/* External inputs (root inport signals with default storage) */
ExtU_ProcessCommands_T ProcessCommands_U;

/* External outputs (root outports fed by signals with default storage) */
ExtY_ProcessCommands_T ProcessCommands_Y;

/* Real-time model */
RT_MODEL_ProcessCommands_T ProcessCommands_M_;
RT_MODEL_ProcessCommands_T *const ProcessCommands_M = &ProcessCommands_M_;

/* Forward declaration for local functions */
static void ProcessCommands_Process_Buttons(void);
static void Pr_chartstep_c1_ProcessCommands(void);

/* Function for Chart: '<Root>/ProcessCommands' */
static void ProcessCommands_Process_Buttons(void)
{
  /* During 'Process_Buttons': '<S1>:7' */
  switch (ProcessCommands_DW.is_Process_Buttons) {
   case ProcessCommands_IN_AMPM:
    /* Inport: '<Root>/Enter' */
    /* During 'AMPM': '<S1>:21' */
    if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
        (ProcessCommands_U.Enter == 1.0)) {
      /* Outport: '<Root>/Time' incorporates:
       *  Outport: '<Root>/Reset'
       */
      /* Transition: '<S1>:54' */
      if (ProcessCommands_Y.Time > 100.0) {
        /* Transition: '<S1>:55' */
        ProcessCommands_Y.Time -= 100.0;
        ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_AMPM;
        ProcessCommands_DW.temporalCounter_i1 = 0U;

        /* Entry 'AMPM': '<S1>:21' */
        ProcessCommands_Y.Reset = 1.0;
      } else {
        /* Transition: '<S1>:53' */
        ProcessCommands_Y.Time += 100.0;
        ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_AMPM;
        ProcessCommands_DW.temporalCounter_i1 = 0U;

        /* Entry 'AMPM': '<S1>:21' */
        ProcessCommands_Y.Reset = 1.0;
      }
    } else {
      if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
          (ProcessCommands_DW.temporalCounter_i1 >= 45)) {
        /* Transition: '<S1>:56' */
        /* Entry Internal 'Process_Buttons': '<S1>:7' */
        /* Transition: '<S1>:31' */
        ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

        /* Outport: '<Root>/SetClock' */
        /* Entry 'Start': '<S1>:5' */
```

```
    ProcessCommands_Y.SetClock = 0.0;
   }
  }
  break;

 case ProcessCommands_IN_Hour1:
  /* Inport: '<Root>/PushButton' */
  /* During 'Hour1': '<S1>:9' */
  if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
      (ProcessCommands_U.PushButton == -1.0)) {
   /* Transition: '<S1>:49' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Wait3;
   ProcessCommands_DW.temporalCounter_i1 = 0U;
  } else {
   /* Outport: '<Root>/Reset' */
   ProcessCommands_Y.Reset = 1.0;
  }
  break;

 case ProcessCommands_IN_Hour2:
  /* Inport: '<Root>/PushButton' */
  /* During 'Hour2': '<S1>:10' */
  if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
      (ProcessCommands_U.PushButton == -1.0)) {
   /* Transition: '<S1>:28' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Wait2;
   ProcessCommands_DW.temporalCounter_i1 = 0U;
  } else {
   /* Outport: '<Root>/Reset' */
   ProcessCommands_Y.Reset = 1.0;
  }
  break;

 case ProcessCommands_IN_Minutes1:
  /* Inport: '<Root>/Enter' */
  /* During 'Minutes1': '<S1>:11' */
  if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
      (ProcessCommands_U.Enter == 1.0)) {
   /* Transition: '<S1>:29' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Output_Time;

   /* Outport: '<Root>/Time' */
   /* Entry 'Output_Time': '<S1>:3' */
   ProcessCommands_Y.Time = (10.0 * ProcessCommands_DW.m2 +
     ProcessCommands_DW.m1) / 100.0 + (10.0 * ProcessCommands_DW.h2 +
     ProcessCommands_DW.h1);

   /* Outport: '<Root>/SetClock' */
   ProcessCommands_Y.SetClock = 1.0;
  }
  break;

 case ProcessCommands_IN_Minutes2:
  /* Inport: '<Root>/PushButton' */
  /* During 'Minutes2': '<S1>:8' */
```

```
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
     (ProcessCommands_U.PushButton == -1.0)) {
   /* Transition: '<S1>:50' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Wait4;
   ProcessCommands_DW.temporalCounter_i1 = 0U;
 } else {
   /* Outport: '<Root>/Reset' */
   ProcessCommands_Y.Reset = 1.0;
 }
 break;

case ProcessCommands_IN_Output_Time:
 /* During 'Output_Time': '<S1>:3' */
 /* Transition: '<S1>:33' */
 ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_AMPM;
 ProcessCommands_DW.temporalCounter_i1 = 0U;

 /* Outport: '<Root>/Reset' */
 /* Entry 'AMPM': '<S1>:21' */
 ProcessCommands_Y.Reset = 1.0;
 break;

case ProcessCommands_IN_SetTemp:
 /* During 'SetTemp': '<S1>:15' */
 if (ProcessCommands_DW.Zone == 2.0) {
   /* Outport: '<Root>/SetTemps' */
   /* Transition: '<S1>:41' */
   ProcessCommands_Y.SetTemps[1] = 10.0 * ProcessCommands_DW.m2 +
     ProcessCommands_DW.m1;

   /* Entry Internal 'Process_Buttons': '<S1>:7' */
   /* Transition: '<S1>:31' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

   /* Outport: '<Root>/SetClock' */
   /* Entry 'Start': '<S1>:5' */
   ProcessCommands_Y.SetClock = 0.0;
 } else {
   if (ProcessCommands_DW.Zone == 1.0) {
     /* Outport: '<Root>/SetTemps' */
     /* Transition: '<S1>:40' */
     ProcessCommands_Y.SetTemps[0] = 10.0 * ProcessCommands_DW.m2 +
       ProcessCommands_DW.m1;

     /* Entry Internal 'Process_Buttons': '<S1>:7' */
     /* Transition: '<S1>:31' */
     ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

     /* Outport: '<Root>/SetClock' */
     /* Entry 'Start': '<S1>:5' */
     ProcessCommands_Y.SetClock = 0.0;
   }
 }
 break;
```

```
case ProcessCommands_IN_SetTime:
 /* Inport: '<Root>/PushButton' */
 /* During 'SetTime': '<S1>:16' */
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_tic) &&
     (ProcessCommands_U.PushButton >= 0.0)) {
   /* Transition: '<S1>:30' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Hour2;

   /* Entry 'Hour2': '<S1>:10' */
   ProcessCommands_DW.h2 = ProcessCommands_U.PushButton;
 } else {
   if ((ProcessCommands_DW.sfEvent == ProcessComman_event_temperature) ||
       (ProcessCommands_DW.sfEvent == ProcessCommands_event_run)) {
     /* Transition: '<S1>:47' */
     /* Transition: '<S1>:52' */
     ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

     /* Outport: '<Root>/SetClock' */
     /* Entry 'Start': '<S1>:5' */
     ProcessCommands_Y.SetClock = 0.0;
   }
 }
 break;

case ProcessCommands_IN_Start_h:
 /* During 'Start': '<S1>:5' */
 if (ProcessCommands_DW.sfEvent == ProcessCommands_event_time) {
   /* Transition: '<S1>:34' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_SetTime;
 } else {
   if (ProcessCommands_DW.sfEvent == ProcessComman_event_temperature) {
     /* Transition: '<S1>:35' */
     ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Temperature;
   }
 }
 break;

case ProcessCommands_IN_Temperature:
 /* Inport: '<Root>/Enter' incorporates:
  *  Inport: '<Root>/PushButton'
  */
 /* During 'Temperature': '<S1>:14' */
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
     (ProcessCommands_U.Enter == 1.0)) {
   /* Transition: '<S1>:39' */
   if (ProcessCommands_DW.Zone == 1.0) {
     /* Transition: '<S1>:42' */
     ProcessCommands_DW.Zone = 2.0;

     /* Outport: '<Root>/Reset' */
     ProcessCommands_Y.Reset = 1.0;

     /* Entry Internal 'Process_Buttons': '<S1>:7' */
     /* Transition: '<S1>:31' */
     ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;
```

```
    /* Outport: '<Root>/SetClock' */
    /* Entry 'Start': '<S1>:5' */
    ProcessCommands_Y.SetClock = 0.0;
   } else {
    /* Transition: '<S1>:43' */
    ProcessCommands_DW.Zone = 1.0;

    /* Outport: '<Root>/Reset' */
    ProcessCommands_Y.Reset = 1.0;

    /* Entry Internal 'Process_Buttons': '<S1>:7' */
    /* Transition: '<S1>:31' */
    ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

    /* Outport: '<Root>/SetClock' */
    /* Entry 'Start': '<S1>:5' */
    ProcessCommands_Y.SetClock = 0.0;
   }
  } else if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_run) ||
             (ProcessCommands_DW.sfEvent == ProcessCommands_event_time)) {
   /* Transition: '<S1>:51' */
   /* Transition: '<S1>:46' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

   /* Outport: '<Root>/SetClock' */
   /* Entry 'Start': '<S1>:5' */
   ProcessCommands_Y.SetClock = 0.0;
  } else {
   if ((ProcessCommands_U.Enter == -1.0) && ((ProcessCommands_DW.sfEvent ==
        ProcessCommands_event_tic) && (ProcessCommands_U.PushButton >= 0.0)))
   {
    /* Transition: '<S1>:44' */
    /* Transition: '<S1>:36' */
    ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Tens1;

    /* Inport: '<Root>/PushButton' */
    /* Entry 'Tens1': '<S1>:12' */
    ProcessCommands_DW.m2 = ProcessCommands_U.PushButton;
   }
  }
  break;

 case ProcessCommands_IN_Tens1:
  /* Inport: '<Root>/PushButton' */
  /* During 'Tens1': '<S1>:12' */
  if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
      (ProcessCommands_U.PushButton == -1.0)) {
   /* Transition: '<S1>:45' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Wait1;
   ProcessCommands_DW.temporalCounter_i1 = 0U;
  } else {
   /* Outport: '<Root>/Reset' */
   ProcessCommands_Y.Reset = 1.0;
  }
```

```
 break;

case ProcessCommands_IN_Tens2:
 /* Inport: '<Root>/Enter' */
 /* During 'Tens2': '<S1>:13' */
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
     (ProcessCommands_U.Enter == 1.0)) {
   /* Outport: '<Root>/Reset' */
   /* Transition: '<S1>:38' */
   ProcessCommands_Y.Reset = 1.0;
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_SetTemp;
 }
 break;

case ProcessCommands_IN_Wait1:
 /* Inport: '<Root>/PushButton' */
 /* During 'Wait1': '<S1>:17' */
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
     (ProcessCommands_U.PushButton >= 0.0)) {
   /* Transition: '<S1>:37' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Tens2;

   /* Entry 'Tens2': '<S1>:13' */
   ProcessCommands_DW.m1 = ProcessCommands_U.PushButton;
 } else {
   if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
       (ProcessCommands_DW.temporalCounter_i1 >= 45)) {
     /* Transition: '<S1>:60' */
     /* Entry Internal 'Process_Buttons': '<S1>:7' */
     /* Transition: '<S1>:31' */
     ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

     /* Outport: '<Root>/SetClock' */
     /* Entry 'Start': '<S1>:5' */
     ProcessCommands_Y.SetClock = 0.0;
   }
 }
 break;

case ProcessCommands_IN_Wait2:
 /* During 'Wait2': '<S1>:18' */
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
     (ProcessCommands_DW.temporalCounter_i1 >= 45)) {
   /* Transition: '<S1>:57' */
   /* Entry Internal 'Process_Buttons': '<S1>:7' */
   /* Transition: '<S1>:31' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

   /* Outport: '<Root>/SetClock' */
   /* Entry 'Start': '<S1>:5' */
   ProcessCommands_Y.SetClock = 0.0;
 } else {
   if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
       (ProcessCommands_U.PushButton >= 0.0)) {
     /* Transition: '<S1>:48' */
```

```
    ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Hour1;

    /* Inport: '<Root>/PushButton' */
    /* Entry 'Hour1': '<S1>:9' */
    ProcessCommands_DW.h1 = ProcessCommands_U.PushButton;
   }
  }
  break;

 case ProcessCommands_IN_Wait3:
  /* During 'Wait3': '<S1>:19' */
  if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
      (ProcessCommands_DW.temporalCounter_i1 >= 45)) {
   /* Transition: '<S1>:58' */
   /* Entry Internal 'Process_Buttons': '<S1>:7' */
   /* Transition: '<S1>:31' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

   /* Outport: '<Root>/SetClock' */
   /* Entry 'Start': '<S1>:5' */
   ProcessCommands_Y.SetClock = 0.0;
  } else {
   if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
       (ProcessCommands_U.PushButton >= 0.0)) {
    /* Transition: '<S1>:32' */
    ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Minutes2;

    /* Inport: '<Root>/PushButton' */
    /* Entry 'Minutes2': '<S1>:8' */
    ProcessCommands_DW.m2 = ProcessCommands_U.PushButton;
   }
  }
  break;

 case ProcessCommands_IN_Wait4:
  /* During 'Wait4': '<S1>:20' */
  if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
      (ProcessCommands_DW.temporalCounter_i1 >= 45)) {
   /* Transition: '<S1>:59' */
   /* Entry Internal 'Process_Buttons': '<S1>:7' */
   /* Transition: '<S1>:31' */
   ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

   /* Outport: '<Root>/SetClock' */
   /* Entry 'Start': '<S1>:5' */
   ProcessCommands_Y.SetClock = 0.0;
  } else {
   if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_next) &&
       (ProcessCommands_U.PushButton >= 0.0)) {
    /* Transition: '<S1>:23' */
    ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Minutes1;

    /* Inport: '<Root>/PushButton' */
    /* Entry 'Minutes1': '<S1>:11' */
    ProcessCommands_DW.m1 = ProcessCommands_U.PushButton;
```

```
        /* Outport: '<Root>/Reset' */
        ProcessCommands_Y.Reset = 1.0;
       }
      }
      break;

     default:
      /* Unreachable state, for coverage only */
      ProcessCommands_DW.is_Process_Buttons = ProcessComma_IN_NO_ACTIVE_CHILD;
      break;
    }
  }

/* Function for Chart: '<Root>/ProcessCommands' */
static void Pr_chartstep_c1_ProcessCommands(void)
{
  int32_T b_previousEvent;

  /* Chart: '<Root>/ProcessCommands' incorporates:
   *  Inport: '<Root>/MeasuredTemps'
   *  Inport: '<Root>/Setting'
   *  Outport: '<Root>/Command'
   *  Outport: '<Root>/Reset'
   *  Outport: '<Root>/SetTemps'
   */
  /* During: ProcessCommands */
  if (ProcessCommands_DW.is_active_c1_ProcessCommands == 0U) {
    /* Entry: ProcessCommands */
    ProcessCommands_DW.is_active_c1_ProcessCommands = 1U;

    /* Entry Internal: ProcessCommands */
    ProcessCommands_DW.is_active_Interactions = 1U;

    /* Entry Internal 'Interactions': '<S1>:6' */
    /* Transition: '<S1>:22' */
    ProcessCommands_DW.is_Interactions = ProcessCommands_IN_Start;
    ProcessCommands_DW.is_active_Process_Buttons = 1U;

    /* Entry Internal 'Process_Buttons': '<S1>:7' */
    /* Transition: '<S1>:31' */
    ProcessCommands_DW.is_Process_Buttons = ProcessCommands_IN_Start_h;

    /* Outport: '<Root>/SetClock' */
    /* Entry 'Start': '<S1>:5' */
    ProcessCommands_Y.SetClock = 0.0;
  } else {
    if (ProcessCommands_DW.is_active_Interactions != 0U) {
      /* During 'Interactions': '<S1>:6' */
      switch (ProcessCommands_DW.is_Interactions) {
       case ProcessCommands_IN_Output:
        /* During 'Output': '<S1>:2' */
        /* Transition: '<S1>:24' */
        ProcessCommands_DW.is_Interactions = ProcessCommands_IN_Start;
        break;
```

```
case ProcessCommands_IN_Start:
 /* During 'Start': '<S1>:1' */
 if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_tic) &&
     (ProcessCommands_U.Setting == 1.0)) {
   /* Transition: '<S1>:25' */
   ProcessCommands_DW.is_Interactions = ProcessComma_IN_NO_ACTIVE_CHILD;
   b_previousEvent = ProcessCommands_DW.sfEvent;
   ProcessCommands_DW.sfEvent = ProcessCommands_event_run;
   if (ProcessCommands_DW.is_active_Process_Buttons != 0U) {
     ProcessCommands_Process_Buttons();
   }

   ProcessCommands_DW.sfEvent = b_previousEvent;
   ProcessCommands_DW.is_Interactions = ProcessCommands_IN_Output;

   /* Entry 'Output': '<S1>:2' */
   ProcessCommands_Y.Command[0] = ProcessCommands_Y.SetTemps[0] -
     ProcessCommands_U.MeasuredTemps[0];
   ProcessCommands_Y.Command[1] = ProcessCommands_Y.SetTemps[1] -
     ProcessCommands_U.MeasuredTemps[1];
   ProcessCommands_Y.Reset = 0.0;
 } else if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_tic) &&
            (ProcessCommands_U.Setting == 3.0)) {
   /* Transition: '<S1>:26' */
   ProcessCommands_DW.is_Interactions = ProcessComma_IN_NO_ACTIVE_CHILD;
   b_previousEvent = ProcessCommands_DW.sfEvent;
   ProcessCommands_DW.sfEvent = ProcessComman_event_temperature;
   if (ProcessCommands_DW.is_active_Process_Buttons != 0U) {
     ProcessCommands_Process_Buttons();
   }

   ProcessCommands_DW.sfEvent = b_previousEvent;
   ProcessCommands_DW.is_Interactions = ProcessCommands_IN_Output;

   /* Entry 'Output': '<S1>:2' */
   ProcessCommands_Y.Command[0] = ProcessCommands_Y.SetTemps[0] -
     ProcessCommands_U.MeasuredTemps[0];
   ProcessCommands_Y.Command[1] = ProcessCommands_Y.SetTemps[1] -
     ProcessCommands_U.MeasuredTemps[1];
   ProcessCommands_Y.Reset = 0.0;
 } else {
   if ((ProcessCommands_DW.sfEvent == ProcessCommands_event_tic) &&
       (ProcessCommands_U.Setting == 2.0)) {
     /* Transition: '<S1>:27' */
     ProcessCommands_DW.is_Interactions = ProcessComma_IN_NO_ACTIVE_CHILD;
     b_previousEvent = ProcessCommands_DW.sfEvent;
     ProcessCommands_DW.sfEvent = ProcessCommands_event_time;
     if (ProcessCommands_DW.is_active_Process_Buttons != 0U) {
       ProcessCommands_Process_Buttons();
     }

     ProcessCommands_DW.sfEvent = b_previousEvent;
     ProcessCommands_DW.is_Interactions = ProcessCommands_IN_Output;
```

```
          /* Entry 'Output': '<S1>:2' */
          ProcessCommands_Y.Command[0] = ProcessCommands_Y.SetTemps[0] -
            ProcessCommands_U.MeasuredTemps[0];
          ProcessCommands_Y.Command[1] = ProcessCommands_Y.SetTemps[1] -
            ProcessCommands_U.MeasuredTemps[1];
          ProcessCommands_Y.Reset = 0.0;
         }
        }
        break;

       default:
        /* Unreachable state, for coverage only */
        ProcessCommands_DW.is_Interactions = ProcessComma_IN_NO_ACTIVE_CHILD;
        break;
       }
     }

     if (ProcessCommands_DW.is_active_Process_Buttons != 0U) {
       ProcessCommands_Process_Buttons();
     }
   }

   /* End of Chart: '<Root>/ProcessCommands' */
}

/* Model output function */
static void ProcessCommands_output(void)
{
   ZCEventType zcEvent_idx_0;
   ZCEventType zcEvent_idx_1;

   /* Chart: '<Root>/ProcessCommands' incorporates:
    *  TriggerPort: '<S1>/ input events '
    */
   /* Inport: '<Root>/ input events ' */
   zcEvent_idx_0 = rt_ZCFcn(RISING_ZERO_CROSSING,
     &ProcessCommands_PrevZCX.ProcessCommands_Trig_ZCE[0],
     (ProcessCommands_U.inputevents[0]));
   zcEvent_idx_1 = rt_ZCFcn(RISING_ZERO_CROSSING,
     &ProcessCommands_PrevZCX.ProcessCommands_Trig_ZCE[1],
     (ProcessCommands_U.inputevents[1]));
   if ((zcEvent_idx_0 != NO_ZCEVENT) || (zcEvent_idx_1 != NO_ZCEVENT)) {
     /* Gateway: ProcessCommands */
     if ((int8_T)zcEvent_idx_0 == 1) {
       /* Event: '<S1>:80' */
       ProcessCommands_DW.sfEvent = ProcessCommands_event_tic;
       Pr_chartstep_c1_ProcessCommands();
     }

     if ((int8_T)zcEvent_idx_1 == 1) {
       if (ProcessCommands_DW.temporalCounter_i1 < 63U) {
         ProcessCommands_DW.temporalCounter_i1++;
       }

       /* Event: '<S1>:83' */
```

```
      ProcessCommands_DW.sfEvent = ProcessCommands_event_next;
      Pr_chartstep_c1_ProcessCommands();
    }
  }
}

/* Model update function */
static void ProcessCommands_update(void)
{
  /* Update absolute time for base rate */
  /* The "clockTick0" counts the number of times the code of this task has
   * been executed. The absolute time is the multiplication of "clockTick0"
   * and "Timing.stepSize0". Size of "clockTick0" ensures timer will not
   * overflow during the application lifespan selected.
   * Timer of this task consists of two 32 bit unsigned integers.
   * The two integers represent the low bits Timing.clockTick0 and the high bits
   * Timing.clockTickH0. When the low bit overflows to 0, the high bits increment.
   */
  if (!(++ProcessCommands_M->Timing.clockTick0)) {
    ++ProcessCommands_M->Timing.clockTickH0;
  }

  ProcessCommands_M->Timing.t[0] = ProcessCommands_M->Timing.clockTick0 *
    ProcessCommands_M->Timing.stepSize0 + ProcessCommands_M->Timing.clockTickH0 *
    ProcessCommands_M->Timing.stepSize0 * 4294967296.0;
}

/* Model initialize function */
static void ProcessCommands_initialize(void)
{
  ProcessCommands_PrevZCX.ProcessCommands_Trig_ZCE[0] = UNINITIALIZED_ZCSIG;
  ProcessCommands_PrevZCX.ProcessCommands_Trig_ZCE[1] = UNINITIALIZED_ZCSIG;

  /* SystemInitialize for Chart: '<Root>/ProcessCommands' */
  ProcessCommands_DW.is_active_Interactions = 0U;
  ProcessCommands_DW.is_Interactions = ProcessComma_IN_NO_ACTIVE_CHILD;
  ProcessCommands_DW.is_active_Process_Buttons = 0U;
  ProcessCommands_DW.is_Process_Buttons = ProcessComma_IN_NO_ACTIVE_CHILD;
  ProcessCommands_DW.temporalCounter_i1 = 0U;
  ProcessCommands_DW.is_active_c1_ProcessCommands = 0U;
  ProcessCommands_DW.m1 = 0.0;
  ProcessCommands_DW.m2 = 0.0;
  ProcessCommands_DW.h1 = 0.0;
  ProcessCommands_DW.h2 = 0.0;
  ProcessCommands_DW.Zone = 1.0;

  /* SystemInitialize for Outport: '<Root>/SetClock' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.SetClock = 0.0;

  /* SystemInitialize for Outport: '<Root>/Time' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.Time = 1.0;
```

```
  /* SystemInitialize for Outport: '<Root>/Reset' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.Reset = 0.0;

  /* SystemInitialize for Outport: '<Root>/SetTemps' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.SetTemps[0] = 70.0;

  /* SystemInitialize for Outport: '<Root>/Command' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.Command[0] = 0.0;

  /* SystemInitialize for Outport: '<Root>/SetTemps' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.SetTemps[1] = 70.0;

  /* SystemInitialize for Outport: '<Root>/Command' incorporates:
   *  Chart: '<Root>/ProcessCommands'
   */
  ProcessCommands_Y.Command[1] = 0.0;
}

/* Model terminate function */
static void ProcessCommands_terminate(void)
{
  /* (no terminate code required) */
}

/*========================================================================*
 * Start of Classic call interface                                        *
 *========================================================================*/
void MdlOutputs(int_T tid)
{
  ProcessCommands_output();
  UNUSED_PARAMETER(tid);
}

void MdlUpdate(int_T tid)
{
  ProcessCommands_update();
  UNUSED_PARAMETER(tid);
}

void MdlInitializeSizes(void)
{
}

void MdlInitializeSampleTimes(void)
{
}
```

```
void MdlInitialize(void)
{
}

void MdlStart(void)
{
  ProcessCommands_initialize();
}

void MdlTerminate(void)
{
  ProcessCommands_terminate();
}

/* Registration function */
RT_MODEL_ProcessCommands_T *ProcessCommands(void)
{
  /* Registration code */

  /* initialize non-finites */
  rt_InitInfAndNaN(sizeof(real_T));

  /* initialize real-time model */
  (void) memset((void *)ProcessCommands_M, 0,
                sizeof(RT_MODEL_ProcessCommands_T));

  /* Initialize timing info */
  {
    int_T *mdlTsMap = ProcessCommands_M->Timing.sampleTimeTaskIDArray;
    mdlTsMap[0] = 0;
    ProcessCommands_M->Timing.sampleTimeTaskIDPtr = (&mdlTsMap[0]);
    ProcessCommands_M->Timing.sampleTimes =
      (&ProcessCommands_M->Timing.sampleTimesArray[0]);
    ProcessCommands_M->Timing.offsetTimes =
      (&ProcessCommands_M->Timing.offsetTimesArray[0]);

    /* task periods */
    ProcessCommands_M->Timing.sampleTimes[0] = (1728.0);

    /* task offsets */
    ProcessCommands_M->Timing.offsetTimes[0] = (0.0);
  }

  rtmSetTPtr(ProcessCommands_M, &ProcessCommands_M->Timing.tArray[0]);

  {
    int_T *mdlSampleHits = ProcessCommands_M->Timing.sampleHitArray;
    mdlSampleHits[0] = 1;
    ProcessCommands_M->Timing.sampleHits = (&mdlSampleHits[0]);
  }

  rtmSetTFinal(ProcessCommands_M, 86400.0);
  ProcessCommands_M->Timing.stepSize0 = 1728.0;
```

```
/* Setup for data logging */
{
  static RTWLogInfo rt_DataLoggingInfo;
  rt_DataLoggingInfo.loggingInterval = NULL;
  ProcessCommands_M->rtwLogInfo = &rt_DataLoggingInfo;
}

/* Setup for data logging */
{
  rtliSetLogXSignalInfo(ProcessCommands_M->rtwLogInfo, (NULL));
  rtliSetLogXSignalPtrs(ProcessCommands_M->rtwLogInfo, (NULL));
  rtliSetLogT(ProcessCommands_M->rtwLogInfo, "tout");
  rtliSetLogX(ProcessCommands_M->rtwLogInfo, "");
  rtliSetLogXFinal(ProcessCommands_M->rtwLogInfo, "");
  rtliSetLogVarNameModifier(ProcessCommands_M->rtwLogInfo, "rt_");
  rtliSetLogFormat(ProcessCommands_M->rtwLogInfo, 0);
  rtliSetLogMaxRows(ProcessCommands_M->rtwLogInfo, 1000);
  rtliSetLogDecimation(ProcessCommands_M->rtwLogInfo, 1);

  /*
   * Set pointers to the data and signal info for each output
   */
  {
    static void * rt_LoggedOutputSignalPtrs[] = {
      &ProcessCommands_Y.SetClock,
      &ProcessCommands_Y.Time,
      &ProcessCommands_Y.Reset,
      &ProcessCommands_Y.SetTemps[0],
      &ProcessCommands_Y.Command[0]
    };

    rtliSetLogYSignalPtrs(ProcessCommands_M->rtwLogInfo, ((LogSignalPtrsType)
      rt_LoggedOutputSignalPtrs));
  }

  {
    static int_T rt_LoggedOutputWidths[] = {
      1,
      1,
      1,
      2,
      2
    };

    static int_T rt_LoggedOutputNumDimensions[] = {
      1,
      1,
      1,
      1,
      1
    };

    static int_T rt_LoggedOutputDimensions[] = {
      1,
```

```
  1,
  1,
  2,
  2
};

static boolean_T rt_LoggedOutputIsVarDims[] = {
  0,
  0,
  0,
  0,
  0
};

static void* rt_LoggedCurrentSignalDimensions[] = {
  (NULL),
  (NULL),
  (NULL),
  (NULL),
  (NULL)
};

static int_T rt_LoggedCurrentSignalDimensionsSize[] = {
  4,
  4,
  4,
  4,
  4
};

static BuiltInDTypeId rt_LoggedOutputDataTypeIds[] = {
  SS_DOUBLE,
  SS_DOUBLE,
  SS_DOUBLE,
  SS_DOUBLE,
  SS_DOUBLE
};

static int_T rt_LoggedOutputComplexSignals[] = {
  0,
  0,
  0,
  0,
  0
};

static RTWPreprocessingFcnPtr rt_LoggingPreprocessingFcnPtrs[] = {
  (NULL),
  (NULL),
  (NULL),
  (NULL),
  (NULL)
};

static const char_T *rt_LoggedOutputLabels[] = {
```

```
  "SetClock",
  "Time",
  "Reset",
  "SetTemps",
  "Command" };

static const char_T *rt_LoggedOutputBlockNames[] = {
  "ProcessCommands/SetClock",
  "ProcessCommands/Time",
  "ProcessCommands/Reset",
  "ProcessCommands/SetTemps",
  "ProcessCommands/Command" };

static RTWLogDataTypeConvert rt_RTWLogDataTypeConvert[] = {
  { 0, SS_DOUBLE, SS_DOUBLE, 0, 0, 0, 1.0, 0, 0.0 },

  { 0, SS_DOUBLE, SS_DOUBLE, 0, 0, 0, 1.0, 0, 0.0 },

  { 0, SS_DOUBLE, SS_DOUBLE, 0, 0, 0, 1.0, 0, 0.0 },

  { 0, SS_DOUBLE, SS_DOUBLE, 0, 0, 0, 1.0, 0, 0.0 },

  { 0, SS_DOUBLE, SS_DOUBLE, 0, 0, 0, 1.0, 0, 0.0 }
};

static RTWLogSignalInfo rt_LoggedOutputSignalInfo[] = {
  {
    5,
    rt_LoggedOutputWidths,
    rt_LoggedOutputNumDimensions,
    rt_LoggedOutputDimensions,
    rt_LoggedOutputIsVarDims,
    rt_LoggedCurrentSignalDimensions,
    rt_LoggedCurrentSignalDimensionsSize,
    rt_LoggedOutputDataTypeIds,
    rt_LoggedOutputComplexSignals,
    (NULL),
    rt_LoggingPreprocessingFcnPtrs,

    { rt_LoggedOutputLabels },
    (NULL),
    (NULL),
    (NULL),

    { rt_LoggedOutputBlockNames },

    { (NULL) },
    (NULL),
    rt_RTWLogDataTypeConvert
  }
};

rtliSetLogYSignalInfo(ProcessCommands_M->rtwLogInfo,
                      rt_LoggedOutputSignalInfo);
```

```
    /* set currSigDims field */
    rt_LoggedCurrentSignalDimensions[0] = &rt_LoggedOutputWidths[0];
    rt_LoggedCurrentSignalDimensions[1] = &rt_LoggedOutputWidths[1];
    rt_LoggedCurrentSignalDimensions[2] = &rt_LoggedOutputWidths[2];
    rt_LoggedCurrentSignalDimensions[3] = &rt_LoggedOutputWidths[3];
    rt_LoggedCurrentSignalDimensions[4] = &rt_LoggedOutputWidths[4];
  }

  rtliSetLogY(ProcessCommands_M->rtwLogInfo, "yout");
}

ProcessCommands_M->solverInfoPtr = (&ProcessCommands_M->solverInfo);
ProcessCommands_M->Timing.stepSize = (1728.0);
rtsiSetFixedStepSize(&ProcessCommands_M->solverInfo, 1728.0);
rtsiSetSolverMode(&ProcessCommands_M->solverInfo, SOLVER_MODE_SINGLETASKING);

/* states (dwork) */
ProcessCommands_M->dwork = ((void *) &ProcessCommands_DW);
(void) memset((void *)&ProcessCommands_DW, 0,
              sizeof(DW_ProcessCommands_T));
ProcessCommands_DW.m1 = 0.0;
ProcessCommands_DW.m2 = 0.0;
ProcessCommands_DW.h1 = 0.0;
ProcessCommands_DW.h2 = 0.0;
ProcessCommands_DW.Zone = 0.0;

/* external inputs */
ProcessCommands_M->inputs = (((void*)&ProcessCommands_U));
ProcessCommands_U.Setting = 0.0;
ProcessCommands_U.PushButton = 0.0;
ProcessCommands_U.Enter = 0.0;
ProcessCommands_U.MeasuredTemps[0] = 0.0F;
ProcessCommands_U.MeasuredTemps[1] = 0.0F;
ProcessCommands_U.inputevents[0] = 0.0;
ProcessCommands_U.inputevents[1] = 0.0;

/* external outputs */
ProcessCommands_M->outputs = (&ProcessCommands_Y);
ProcessCommands_Y.SetClock = 0.0;
ProcessCommands_Y.Time = 0.0;
ProcessCommands_Y.Reset = 0.0;
ProcessCommands_Y.SetTemps[0] = 0.0;
ProcessCommands_Y.SetTemps[1] = 0.0;
ProcessCommands_Y.Command[0] = 0.0;
ProcessCommands_Y.Command[1] = 0.0;

/* Initialize Sizes */
ProcessCommands_M->Sizes.numContStates = (0);/* Number of continuous states */
ProcessCommands_M->Sizes.numY = (7); /* Number of model outputs */
ProcessCommands_M->Sizes.numU = (7); /* Number of model inputs */
ProcessCommands_M->Sizes.sysDirFeedThru = (1);/* The model is direct feedthrough */
ProcessCommands_M->Sizes.numSampTimes = (1);/* Number of sample times */
ProcessCommands_M->Sizes.numBlocks = (8);/* Number of blocks */
ProcessCommands_M->Sizes.numBlockIO = (5);/* Number of block outputs */
```

```
  return ProcessCommands_M;
}

/*=========================================================================*
 * End of Classic call interface                                           *
 *=========================================================================*/
```

ДОДАТОК Б

Модель системи автоматизованого керування обігріву двозонного приміщення

УКР.НТУУ “КПІ” ім. І. Сікорського. ТО41206

Аркушів 4

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVII Міжнародної науково-практичної конференції молодих вчених та студентів м. Київ, 23-26 квітня 2019 року,

ТОМ 2

Київ- 2019

УДК 697
Магістрант 1 курсу, гр. ТО-81мп Тарасюта Д.О.
Проф., д.т.н. Волощук В.А.

ОСОБЛИВОСТІ ФУНКЦІОНУВАННЯ "РОЗУМНИХ" ТЕПЛОВИХ МЕРЕЖ

З метою збільшення можливостей утилізації скидної енергії, ширшого залучення відновлювальних джерел енергії, а також зниження втрат енергії у формі теплоти у випадку її транспортування у сфері теплозабезпечення населених пунктів пропонується зниження температурного рівня теплоносіїв. З'явилася концепція четвертого покоління централізованих систем теплопостачання [1], а також концепція низькоексергетичних систем теплозабезпечення [2]. Дані рішення передбачають, зокрема, можливість створення єдиної «розумної» енергетичної системи, що включає в себе «розумні» електричні, теплові, газові та водопровідні мережі для забезпечення синергетичного ефекту, появу просьюмерів (prosumers) – тобто таких «гравців», які одночасно можуть бути як виробниками так і споживачами енергії, а також використання технологій індивідуального управління (supportive technologies) [1, 3].

Отже, роль технологій «smart» у сфері теплозабезпечення будівель зростає.

Метою роботи є огляд особливостей створення та функціонування «розумних» теплових мереж.

Технологічно елементи «розумних» теплових мереж передбачають об'єднання генеруючих установок невеликої потужності в тому числі і на основі відновлювальних джерел енергії, когенераційних установок, сезонних та корокотермінових (від декількох днів до годин) теплових акумуляторів, впровадження нових підходів до приготування гарячої води, інноваційних рішень у системах транспортування (застосування нових матеріалів на основі пластику, теплопроводи типу twin-pipes та triple-pipes).

«Розумні» теплові мережі повинні характеризуватися адаптивністю, саморегулюванням, забезпечувати можливість інтеграції із іншими системами, бути енергоефективними та привабливими як для споживачів, так і для інвесторів [4].

В теперішній час уже створено достатньо багато систем теплозабезпечення із застосуванням принципів «розумних теплових мереж» як на стадії тестування та демонстрування, так і на стадії комерційного впровадження. Спостерігається подальше удосконалення таких технологій [5].

Перелік посилань:

1. Lund H. 4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems/ H. Lund, S. Werner, R. Wiltshire, S. Svendsen, J. E. Thorsen, F. Hvelplund, B. V. Mathiesen // Energy. – 2014. –Vol. 68. – P. 1–11.
2. Hepbasli A. Low exergy (LowEx) heating and cooling systems for sustainable buildings and societies / A. Hepbasli // Renewable and Sustainable Energy Reviews. – 2012. – Vol. 16(1). – P. 73–104.
3. Paiho S. Towards next generation district heating in Finland / S. Paiho, F. Reda // Renewable and Sustainable Energy Reviews. – 2016. –Vol. 65. – P. 915–924.
4. Schmidt A. Smart cities: challenges and opportunities for thermal networks / A. Ralf-Roman Schmidt, B. Olivier Pol, C. Jessen Page// The 14th International Conference of Young Scientists on Energy Issues (CYSENI-2017). Program and Proceedings, Kaunas, Lithuania May, 25–26, 2017. – Kaunas, 2017. – P. II-108.
5. Stănişteanu C. Smart thermal grids – a review / C. Stănişteanu // Scientific Bulletin of the Electrical Engineering Faculty . – 2017. – No.1(36).